Self-adaptive differential artificial bee colony algorithm for global optimization problems

Chen, Xu; Tianfield, Huaglory; Li, Kangji

Published in:
Swarm and Evolutionary Computation

DOI:
10.1016/j.swevo.2019.01.003

Publication date:
2019

Document Version
Author accepted manuscript

Self-adaptive differential artificial bee colony algorithm for global optimization problems

Xu Chen[a,*], Huaglory Tianfield[b], Kangji Li[a]

[a] *School of Electrical and Information Engineering, Jiangsu University, Zhenjiang 212013, Jiangsu, China*
[b] *School of Engineering and Built Environment, Glasgow Caledonian University, Glasgow G4 0BA, UK.*

Abstract

Artificial bee colony algorithm (ABC) has attracted wide attention in the recent decade. Although ABC algorithms can achieve good performance on separable problems by optimizing each variable independently, their performances on complex non-separable problems are still unsatisfactory. In this paper, through incorporating multiple differential search strategies and a self-adaptive mechanism within the framework of ABC, we propose a new ABC algorithm, called self-adaptive differential artificial bee colony (sdABC) algorithm. By means of differential search strategies, more variables will be updated each time based on the combination of mutation and crossover. Thus, sdABC has much enhanced ability for solving complex non-separable problems. Our proposed sdABC algorithm is evaluated on 28 benchmarks functions, including both common separable problems and complex non-separable CEC2015 functions. The experimental results show that sdABC can achieve much more desirable results compared with previous ABC algorithms on both separable and non-separable functions. Furthermore, the performance of our sdABC is also very competitive compared with well-established differential evolution and other meta-heuristic algorithms.

Keywords: Artificial bee colony, differential search, self-adaptive search, non-separable problem, meta-heuristic algorithm

1. Introduction

Global optimization problems (GOPs) are widespread in scientific and engineering domains, such as structural design, scheduling, portfolio investment, and power economic dispatch. GOPs are often characterized by non-convexity, discontinuity, non-differentiability and multi-modality [1]. Traditional optimization methods based on mathematical programming may become unusable or unrealizable for these GOPs[2, 3]. In the recent decades, nature-inspired meta-heuristic algorithms (MHAs) have emerged as powerful optimization tools for solving GOPs. Such MHAs include genetic algorithm (GA) [4], differential evolution (DE) [5, 6], particle swarm optimization (PSO) [7, 8], artificial bee colony (ABC) [9, 10], biogeography-based optimization (BBO) [11, 12], teaching-learning-based optimization (TLBO) [13, 14], and artificial raindrop algorithm (ARA) [15].

Among the aforementioned nature-inspired MHAs, ABC has attracted wide attentions in recent years due to its robust performance for solving optimization problems [16]. ABC was firstly introduced by Karaboga, with

*Corresponding author
Email addresses: xuchen@ujs.edu.cn (Xu Chen), H.Tianfield@gcu.ac.uk (Huaglory Tianfield)

the inspiration from the intelligent foraging behavior of honey bee swarm [17]. ABC simulates three kinds of bees, i.e., employed bees, onlookers, and scouts, to search for good food sources. Compared with other MHAs, ABC has the advantages such as simple structure, ease of implementation and robust performance [18]. Therefore, ABC algorithms have been applied to different types of optimization problems, including continuous optimization [18], multi-objective optimization [19, 20], combinatorial optimization [21, 22], data clustering [23], and various real-world domain problems [24, 25].

ABC suffers from the drawbacks like slow convergence and poor exploitation [26]. The reason is that the search strategy used in ABC only updates one variable at a time, which results in slow evolution [27]. To address this issue, improvements to ABC algorithm have been introduced by, e.g., modifying search equations [28, 29, 30], hybridizing with other meta-heuristic algorithms [31, 32, 24] and employing multiple search strategies [33, 34, 35]. While the convergence of these ABC variants have been prominently increased, most of these improved ABC algorithms are still confined to updating one variable at a time. In particular, although this kind of updating strategy may achieve fairly good performance on separable problems by optimizing each variable independently, the performance of these improved ABC algorithms for non-separable problems is still unsatisfactory [36]. This may become a major restriction to ABC algorithms because when given an optimization problem, it is unrealistic to require that the problem is separable. In fact, the benchmark problems such as the CEC competition functions [37] and most of the real-world optimization problems are non-separable. Therefore, it is of great significance if ABC algorithm can be fundamentally enhanced for the complex non-separable problems.

This paper is focused on enhancing ABC for solving complex non-separable problems. Through incorporating differential search strategies and a self-adaptive mechanism into ABC framework, we propose a new ABC algorithm, called self-adaptive differential artificial bee colony (sdABC). Our sdABC employs different search strategies of differential evolution in both employed and onlooker bee updating phases. By means of differential search strategies, more variables are updated each time based on the combination of mutation and crossover. Undoubtedly, this will be very beneficial for the enhancing the ability of ABC in solving complex non-separable problems. Moreover, selection of differential search strategies is calculated by using a probability based self-adaptive mechanism, which can select the most appropriate search strategies for the sdABC.

The reminder of the paper is arranged as follows. Section 2 presents a literature review of the ABC algorithms. Section 3 develops our proposed sdABC algorithm. In Section 4, comprehensive experiments are carried out to evaluate the proposed algorithm. Finally, Section 5 draws up conclusions.

2. Literature review

Since the advent of ABC, many improvements have been introduced to enhance the performance of ABC, resulting in numerous ABC variants.

(1) **Modifications to the search equations of ABC**.

The search equation of basic ABC only updates one variable at a time, and thus the exploitation in the basic ABC is quite week. Therefore, many new search equations are introduced to enhance the exploitation

ability. Zhu and Kwong [26] incorporated the information of global best solution into the search equation and proposed a gbest-guided ABC (GABC) . Akay and Karaboga [38] introduced modification rate (MR) and scaling factor (SF) to control the frequency and the magnitude of perturbation, respectively, and proposed a modified ABC (MABC). Gao et al. [29] presented two modified search strategies for ABC, which is inspired from the mutation operators "best/1" and "best/2" of differential evolution. Sharma et al. [39] incorporated the opposition based learning strategy and Levy flight random walk into ABC, and put forward an opposition based Levy flight ABC (OBLFABC). Kiran et al. [40] added the directional information to the search equation, and presented a directed ABC algorithm (dABC). Gao et al. [30] applied the orthogonal experimental design to form an orthogonal learning strategy for ABC, and proposed a new ABC algorithm based on modified search equation and orthogonal learning (OCABC). Zhou et al. [41] adopted Gaussian bare-bones search equation in the onlooker bee phase, and proposed Gaussian bare-bones ABC (GBABC). Cui et al. [18] developed a ranking-based adaptive ABC (ARABC), in which DE mutation operator "rand/1" as the search equation, and the parent food sources used in the search equation are chosen based on their rankings. Yu et al.[42] proposed an adaptive ABC (AABC), by using a novel greedy position update strategy and an adaptive control scheme for adjusting the greediness degree. Song et al. [43] developed a novel search equation by using the information of objective function value, and the new search equation can efficiently adjust the step-size adaptively. Cui et al. [28] proposed a depth-first search ABC with elite-guided search equation (DFSABC_elite), by incorporating the information of elite solutions into solution search equations.

(2) **Hybrid ABC with other meta-heuristic algorithms**.

Hybrid algorithms based on two or more meta-heuristics may maximize individual algorithms advantages and enhance the optimization performance. This is also the case for hybrid ABC algorithms by combining ABC with other meta-heuristic algorithms. Chen et al. [32] proposed a simulated annealing based ABC (SAABC) by modifying the employed bees searching process using simulated annealing algorithm. Xiang et al. [44] proposed a particle swarm like multi-elitist ABC (PS-MEABC), in which the global best solution and an elitist randomly selected from the elitist archive are used to modify the parameters of food sources. Jadon et al. [45] presented a hybridization of ABC and DE algorithms called HABCDE, in which the onlooker bee phase is improved by evolutionary operators of basic DE process for faster convergence. Liang et al. [27] put forward an enhanced ABC algorithm with adaptive differential operators (ABCADE) in which the differential operators are adopted to generate new solutions with an increasing probability, and the associated parameters are adaptively adjusted based on Gaussian distribution. Wu et al. [46] proposed a hybrid harmony search with ABC algorithm (HHSABC) for solving global numerical optimization problems. Cai et al. [31] integrated ABC with biogeography-based optimization algorithm, and proposed ABC-BBO algorithm for constrained mechanical design problems. Zhang et al. [47] developed modified ABC by embedding grenade explosion method, which makes employed bees or onlooker bees search the food source using the optimal search dimension. Chen et al. [24] developed a hybrid teaching-learning-based artificial bee colony (TLABC) by introducing the teacher and learner operators into the bee phases of ABC. Wang and Yi [48] presented a robust optimization algorithm called KHABC based on hybridization of krill herd and ABC.

(3) **Employments of multiple search strategies with ABC.**

The idea of multiple search strategies has been introduced to ABC, and several multiple-strategy-based ABC algorithms have been developed in recent years. In these algorithms, different search strategies are used for different food sources, which can enhance the universality and robust performance of ABC. Wang et al. [34] proposed a multi-strategy ensemble ABC (MEABC), in which a pool of strategies is constructed by three distinct search strategies to compete to produce offspring. Gao et al. [49] developed an ABC with multiple search strategies (MuABC) . In MuABC, employed bee and onlooker bee each generate three candidate solutions by using three different search strategies, and the best solution is selected to compete with the old solution. Kiran et al. [33] put forward an ABC algorithm with variable search strategy (ABCVSS). This method uses five different search strategies and the selection probabilities of these strategies are computed based on the number of their successes. Harfouchi et al. [50] designed a modified cooperative learning ABC (mCLABC), which partitions the populations into three subgroups, and each subgroup uses two different solution search equations. Xue et al. [35] presented a self-adaptive ABC based on global best solution (SABCGB). SABCGB employed three search strategies using the global best solution, and a self-adaptive mechanism is also used to select search strategies.

Besides the above ABC algorithms, El-Abd [51]proposed a generalized opposition-based ABC (GOABC) algorithm by introducing the concept of generalized opposition-based learning. Cui et al. [36] proposed an enhanced ABC algorithm with dual-population framework.

3. Development of ABC with self-adaptive differential search strategies

3.1. ABC preliminaries

ABC is inspired by the cooperative foraging behaviors of honey bee colony [17]. In ABC, the foraging process of honey bee colony is equivalent to the optimization process of discovering an optimal solution. The location of a food source is regarded as a candidate solution to the optimization problem, and the amount of nectar in each food source denotes the quality of the corresponding solution. A honey bee colony consists of three bee groups: employed bees, onlookers, and scouts. The first half of the colony serve as employed bees, and the other half serve as the onlookers. Employed bees are responsible for searching better food sources and passing the quality information of the food sources to onlookers by waggle dancing. Onlookers select good food sources found by employed bees to strengthen the search for better food sources. When the quality of a food source is not improved over predefined cycles (*limit*), the food source is regarded as exhausted and will be abandoned by its employed bee, and the employed bees become scouts, which seek out new food sources randomly.

In the initialization phase, ABC algorithm first randomly generates SN food sources (or solution) $\boldsymbol{x}_i = (x_{i1}, \cdots, x_{ij}, \cdots, x_{iD}), i = 1, \cdots, SN$ as below:

$$x_{ij} = x_j^{\min} + (x_j^{\max} - x_j^{\min}) \cdot rand_{ij} \tag{1}$$

where x_{ij} represents the j-th element of the i-th solution $\boldsymbol{x}_i$; $x_j^{\min}$ and $x_j^{\max}$ are the j-th dimensional elements

of lower and upper bounds, respectively; $rand_{ij}$ is a random real number uniformly distributed within [0, 1]. The objective function values for all solutions are also evaluated as $f(\boldsymbol{x}_i),\ i = 1, \cdots, SN$.

After the initialization, ABC will enter into three bee updating phases, i.e., employed bee updating, onlooker bee updating and scout bee updating, until the terminated condition is satisfied.

In the employed bee updating phase, each employed bee searches around a unique food source and generates a new food source $\boldsymbol{u}_i = (u_{i1}, u_{i2}, \cdots, u_{iD})$ as below:

$$u_{ij} = x_{ij} + \psi_{ij} \cdot (x_{ij} - x_{kj}) \tag{2}$$

where $i, k \in \{1, 2, \cdots, SN\}$ and $k \neq i$; $j \in \{1, 2, \cdots, D\}$; ψ_{ij} is a random real number within [-1, 1]. If $\boldsymbol{u}_i$ is better than the old food source $\boldsymbol{x}_i$, then $\boldsymbol{x}_i$ will be replaced by $\boldsymbol{u}_i$ and its counter will be set as $counter = 0$. Otherwise, the old one $\boldsymbol{x}_i$ will be kept, and its *counter* will be increased by one.

In the onlooker bee updating phase, employed bees have finished their search tasks and will share their information of food sources to the onlooker bees. Each onlooker bee chooses a food source to carry out further search based on probability p_i, which is calculated as below:

$$p_i = \frac{fit(\boldsymbol{x}_i)}{\sum_{i=1}^{SN} fit(\boldsymbol{x}_i)} \tag{3}$$

where $fit(\boldsymbol{x}_i)$ is the fitness values of food source $\boldsymbol{x}_i$ and is calculated based on the objective function value $f(\boldsymbol{x}_i)$ as below:

$$fit(\boldsymbol{x}_i) = \begin{cases} \frac{1}{1+f(\boldsymbol{x}_i)} & if\ \ f(\boldsymbol{x}_i) \geq 0 \\ 1 + |f(\boldsymbol{x}_i)| & otherwise \end{cases} \tag{4}$$

According to Eq. (3), probability p_i is proportional to the fitness value of the food source. Food sources with larger fitness values will have larger selection probabilities. Each onlooker bee flies to a selected food source $\boldsymbol{x}_s$ which is selected based on roulette method, and then employs Eq. (2) to produce a candidate food source $\boldsymbol{u}_s$. Similarly, if $\boldsymbol{u}_s$ is better than $\boldsymbol{x}_s$, $\boldsymbol{x}_s$ will replace $\boldsymbol{u}_s$ and its *counter* will be set to 0. Otherwise, $\boldsymbol{x}_s$ will be kept, and its *counter* will be increased by one.

In scout bee updating phase, the food source with the highest *counter* is selected. If its *counter* is greater than a predefined *limit* value, this food source will be abandoned, and its employed bee will reset as a scout bee to regenerate a new food source randomly according to Eq. (1). After the new food source is generated, its *counter* is set to 0, and the scout bee become an employed bee.

3.2. Bee updating by multiple differential search strategies

The search strategy of ABC algorithm (i.e., Eq.2) only updates one variable at a time, and the performance of this search strategy is poor for non-separable problems. To enhance the search ability, researchers have proposed modifying the search equations or employing multiple search strategies in the bee updating phases; however, the search strategies used in most of these ABC variants are still confined to updating one variable at a time, which largely restricts the search abilities of the algorithms.

Compared with the bee updating strategy in Eq.(2), differential search strategies from DE algorithm [52, 53] have some different features that make them more efficient for complex non-separable problems. The first is that differential search strategies modify more variables based on the combination of mutation and crossover, and the number of updating variables can also be adjusted by a control parameter called crossover rate. The second is that the step size of differential search is adapted based on the population in the optimization process. In detail, the step size is big at the beginning when the population is diverse, and the step size then decreases as the population converge. Thus, the differential search strategies initially favor exploration then switches to exploitation. The third is that a number of distinct differential search strategies which are competent for different optimization problems have been designed in previous studies, and a combination of distinct differential search strategies with a self-adaptive mechanism can further enhance the robustness of the proposed algorithm. Inspired from this observation, we are incorporating the differential search strategies into ABC to form a self-adaptive differential ABC algorithm.

To design the sdABC algorithm, we will address two key issues: (1) which differential strategies are to be employed, and (2) how the probabilities for selecting these differential strategies are calculated.

We consider three distinct differential search strategies for bee search equations in sdABC, namely "rand/1/bin", "current-to-pbest/1/bin" with an archive, and "current-to-rand/1".

(i) "rand/1/bin"

$$u_{i,j} = \begin{cases} x_{r_1,j} + F \cdot (x_{r_2,j} - x_{r_3,j}), & \text{if } rand \leq CR \ \ or \ \ (j = j_{rand}) \\ x_{i,j}, & \text{otherwise} \end{cases} \tag{5}$$

(ii) "current-to-pbest/1/bin"

$$u_{i,j} = \begin{cases} x_{i,j} + F \cdot (x_{pbest,j} - x_{i,j} + x_{r_1,j} - \tilde{x}_{r_2,j}), & \text{if } rand \leq CR \ \ or \ \ (j = j_{rand}) \\ x_{i,j}, & \text{otherwise} \end{cases} \tag{6}$$

(iii) "current-to-rand/1"

$$u_{i,j} = x_{i,j} + L \cdot (x_{r_1,j} - x_{i,j}) + F \cdot (x_{r_2,j} - x_{r_3,j}) \tag{7}$$

where, $x_{i,j}$ and $u_{i,j}$ are the j-th element of $\boldsymbol{x}_i$ and $\boldsymbol{u}_i$, respectively; r_1, r_2 and r_3 are mutually exclusive random integers within $[1, SN]$; F is the scale factor that controls the amplification of the difference vectors; CR is crossover rate; L is a real random number within [0,1]; j_{rand} is a random integer within $[1, D]$; $x_{pbest,j}$ denotes the j-th element of $\boldsymbol{x}_{pbest}$, $\boldsymbol{x}_{pbest}$ is randomly selected from the top $100 * p$ solutions in the present population; and $\tilde{x}_{r_2,j}$ represents the j-th element of $\tilde{\boldsymbol{x}}_{r_2}$, which is randomly chosen from the union of current population and archived inferior solution set.

Remark 1: The "rand/1/bin" is the most robust and common differential strategy that is usually suitable for solving multimodal problems [54]. The "current-to-pbest/1 bin" with an archive is originally proposed in [55], which shows very competitive performance in solving complex optimization problems, especially those with unimodal landscapes. The "current-to-rand/1" is a strategy without the crossover operation, and is

particularly useful for solving rotated problems, as it is rotation-invariant [56]. As these three strategies are complementary to each other, it is useful to combine the best features of different strategies.

Remark 2: We will use the parameter adaptive technique in [55, 57] to adjust scale factor F and crossover probability CR, which are two important parameters of differential strategies. The detail of the parameter adaptive technique is provided in **Appendix 1**.

The search strategy in sdABC algorithm is called multiple-differential-strategy-based bee updating in this paper. As illustrated in Fig.1, at any point of time, each food source will be assigned with one differential search strategy from the pool of strategies, and the food source will be modified based on its corresponding strategy in different bee updating phases.

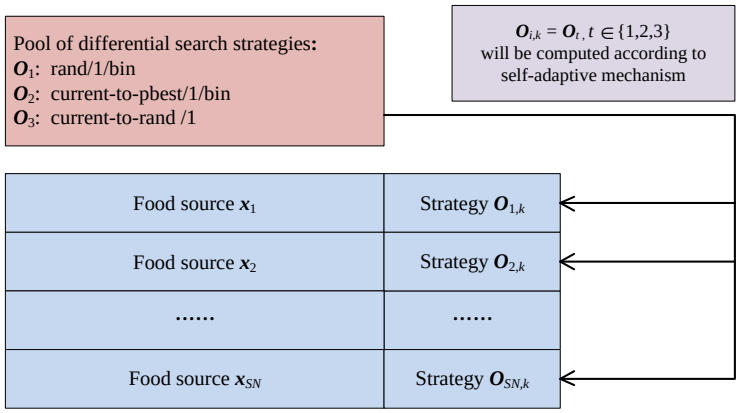


Figure 1: Multiple-differential-strategy-based coding method

3.3. Self-adaptive mechanism

The probabilities for selecting differential search strategies for different food sources are dynamically adjusted in the optimization process. In the initialization stage, the probability for selecting a strategy is set equally as below:

$$pa_k = \frac{1}{K}, k = 1, \cdots, K \tag{8}$$

where pa_k is the probability for selecting the k-th strategy; K is the number of strategies, and here $K = 3$. As the optimization search proceeds, the probabilities for selecting different strategies evolve based on their performance in generating promising solutions. The following self-adaptive mechanism is used to update the selection probabilities.

At the beginning, we use two parameters Δf_k and ΔFes_k to record the accumulated improvement of objective function brought forth by the k-th strategy and accumulated functional evaluations cost by the k-th strategy, respectively. Then, after multiple-differential-strategy-based bee updating phases, the parameters Δf_k and ΔFes_k are used to update the selection probability for k-th strategy as below:

$$pa_k = pa_{\min} + (1 - K \cdot pa_{\min}) \frac{\Delta af_k}{\sum_{i=1}^{3} \Delta af_i}, \quad k = 1, 2, 3 \tag{9}$$

where Δaf_k is the ratio of improvement of objective function to evaluation cost, and computed as $\Delta af_k = \Delta f_k / \Delta Fes_k$; $pa_{\min}$ is the minimum selection probability for each strategy.

The proposed self-adaptive differential ABC (sdABC) approach can be illustrated by the flowchart in Fig.2. The detailed pseudocode of sdABC can be formulated as in **Algorithm 1**. It elaborates how the multiple-differential-strategy-based bee updating and the self-adaptive mechanism are built within the framework of ABC.

Algorithm 1 Self-adaptive differential artificial bee colony algorithm

Stage 1: Initialization
Set $\Delta f_k = 0$, $\Delta Fes_k = 0$, and $pa_k = 1/3$ for each $k = 1, 2, 3$;
for each index $i = 1 \rightarrow SN$ **do**
  Initialize the position of food source $\boldsymbol{x}_i$;
  Calculate the objective function value $f(\boldsymbol{x}_i)$;
  Set $counter(i)$=0;
end for
Stage 2: Assignment of differential search strategies
for each index $i = 1 \rightarrow SN$ **do**
  Select an differential search strategy $\boldsymbol{O}_{i,k}$ from $\{\boldsymbol{O}_1, \boldsymbol{O}_2, \boldsymbol{O}_3\}$ using the roulette method based on probabilities $pa_k,\ k = 1, 2, 3$;
end for
Stage 3: Employed bee phase based on multiple differential strategies
for each index $i = 1 \rightarrow SN$ **do**
  Generate a new candidate solution $\boldsymbol{u}_i$ by differential strategy $\boldsymbol{O}_{i,k}$;
  Calculate the objective function value $f(\boldsymbol{u}_i)$;
  if $f(\boldsymbol{u}_i) < f(\boldsymbol{x}_i)$ **then**
    $\Delta f_k = f(\boldsymbol{x}_i) + (f(\boldsymbol{x}_i) - f(\boldsymbol{u}_i))$, $\Delta Fes_k = \Delta Fes_k + 1$;
    $\boldsymbol{x}_i = \boldsymbol{u}_i$, $counter(i) = 0$;
  else
    $counter(i) = counter(i) + 1$;
  end if
end for
Stage 4: Onlooker bee phase based on multiple differential strategies
Calculate the selection probability p_i using Eq.(3);
for each index $i = 1 \rightarrow SN$ **do**
  Select a solution $\boldsymbol{x}_s$ using the roulette method according to the probability p_i;
  Generate a new candidate solution $\boldsymbol{u}_s$ by differential strategy $\boldsymbol{O}_{s,k}$;
  Calculate the objective function value $f(\boldsymbol{u}_s)$;
  if $f(\boldsymbol{u}_s) < f(\boldsymbol{x}_s)$ **then**
    $\Delta f_k = f(\boldsymbol{x}_s) + (f(\boldsymbol{x}_s) - f(\boldsymbol{u}_s))$, $\Delta Fes_k = \Delta Fes_k + 1$;
    $\boldsymbol{x}_s = \boldsymbol{u}_s$, $counter(s) = 0$;
  else
    $counter(s) = counter(s) + 1$;
  end if
end for
Stage 5: Scout bee updating phase
Find out the solution $\boldsymbol{x}_{\text{Index}}$ with the maximum *counter* value;
if $count$(index) $\geq limit$ **then**
  Replace $\boldsymbol{x}_{\text{index}}$ with a new randomly generated solution;
  Calculate the objective function value $f(\boldsymbol{x}_{\text{Index}})$;
  Set $count$(index) = 0;
end if
Stage 6: Update the selection probabilities using self-adaptive mechanism
$\Delta af_k = \Delta f_k / \Delta Fes_k,\ k = 1, 2, 3$;
$sum = \Delta af_1 + \Delta af_2 + \Delta af_3$;
Calculate the probability $pa_k = pa_{\min} + (1 - K \cdot pa_{\min})(\Delta af_k / sum)$;
Set $\Delta f_k = 0$, $\Delta Fes_k = 0$;
If the halting criterion is not satisfied, go to **Stage 2**; otherwise, output result.

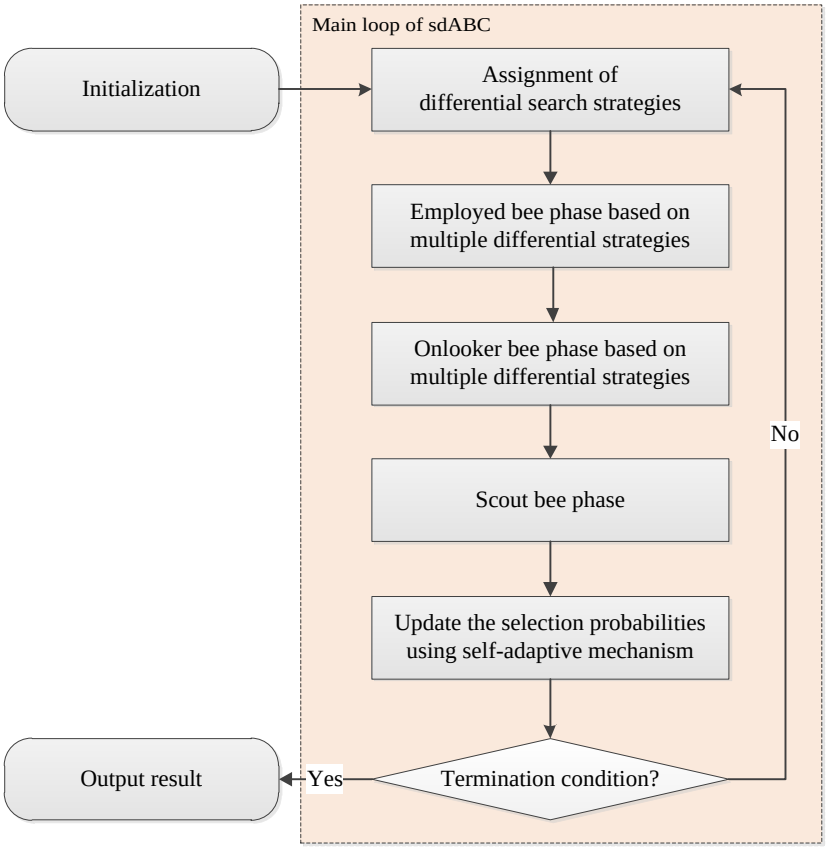


Figure 2: Flowchart of the sdABC algorithm.

Remark 3: In the literature there are a number of ABC algorithms which uses multiple search strategies, such as MEABC[34], SABCGB [35] and ABCVSS [33]. However, our sdABC is different from them in several aspects. Firstly, the search strategies in previous multiple-strategy-based ABC algorithms only update one variable at a time, whereas the search strategies in our sdABC cover diverse differential strategies which modify more variables based on the combination of mutation and crossover. Secondly, the self-adaptive mechanism in our sdABC is designed based on accumulated improvement of objective function value, which is different from previous multiple-strategy-based ABC algorithms. Finally, our sdABC aims at improving ABC for non-separable problems by multiple differential strategies, while previous multiple-strategy-based ABC algorithms focus on improving the convergence of ABC with multiple strategies. With the above differences, we believe that our sdABC will further improve the performance of previous multiple-strategy-based ABC algorithms, especially for complex non-separable problems.

Remark 4: Kiran et al.[40] presented a directed ABC algorithm (dABC) in which directional information is added to ABC for improving convergence and local search capability. Gao et al. [58] proposed an enhanced ABC (EABC) algorithm using more information-based search equations. Both dABC and EABC focus on improving the convergence of ABC by adding either directional information or gbest information, but their search equations still update one variable at a time. Different from dABC and EABC, our sdABC focuses on enhancing ABC for complex non-separable problems by employing self-adaptive differential strategies.

Remark 5: The computational costs of our sdABC mainly involve the following parts: (a) assignment of the differential search strategies (T_A), employed bee updating phase (T_E), onlooker bee updating phase (T_O), scout bee phase(T_S), and (b) selection probabilities updating (T_U). Therefore, the total computational complexity of sdABC is:

$$\begin{aligned} T_{sdABC} &= (T_A + T_E + T_O + T_S + T_U) \cdot G_{\max} \\ &= (O(SN \times K) + O(SN \times D) + O(SN \times D) + O(D) + O(K)) \cdot G_{\max} \\ &= (O(SN \times (2D + K) + D + K)) \cdot G_{\max} \end{aligned} \tag{10}$$

where $G_{\max}$ is the maximal number of generations, $K = 3$ is the number of differential search strategies. According to the operational rules of the symbol O, $T_{sdABC} = (O(SN \times (2D + 3) + D + 3)) \cdot G_{\max} = O(SN \times D \times G_{\max})$.

4. Performance evaluation

We evaluate the performance of the proposed sdABC algorithm. Our experimental studies can be divided into seven cases:

(1) Subsection 4.1 compares sdABC with six single-strategy-based ABC algorithms.

(2) Subsection 4.2 compares sdABC with two ABC and four DE algorithms which are built with multiple search strategies.

(3) Subsection 4.3 compares sdABC with five non-ABC and non-DE algorithms

(4) Subsection 4.4 compares sdABC with six excellent meta-heuristic algorithms which had taken part in the competition on real-parameter single objective optimization at CEC2015.

(5) Subsection 4.5 evaluates the time complexity of sdABC algorithm.

(6) Subsection 4.6 analyzes the search behaviors of self-adaptive differential strategies in sdABC.

(7) Subsection 4.7 applies sdABC to solve the real-world parameter estimation problems of solar cell models.

In total, 28 benchmark functions are employed to evaluate the performance of sdABC [1], as listed in Table 1. These benchmark functions fall into two groups: (1) G1: the 13 widely used conventional functions from Yao et al [59], and (2) G2: the 15 complex functions taken from the whole test suite of CEC2015 [37]. From Table 1, most of conventional functions are separable or partially separable, while all the CEC2015 functions are non-separable.

All algorithms are run 51 times independently to obtain the statistical results, and the maximum number of function evaluations is set as $maxFES = 10000 \times D$. The mean and standard deviations (SD) of the function error value $f(\boldsymbol{x}_{best}) - f(\boldsymbol{x}^*)$ are used to evaluate the optimization performance, where $\boldsymbol{x}_{best}$ is the best solution found by the algorithm in each run and $\boldsymbol{x}^*$ is the true global optimal solution. Moreover, nonparametric statistical tests, including Wilcoxon's rank sum test and Friedman's test, are conducted to establish a reliable statistic view.

[1] The source code of sdABC is available from the first author upon request.

Table 1: Benchmark functions used in performance evaluations

Function	Name	Search Range	Optimal		
Group 1: Conventional functions					
f1	Sphere model	$[-100, 100]^D$	0	Unimodal	Separable
f2	Schwefel's problem 2.22	$[-10, 10]^D$	0	Unimodal	Partially Separable
f3	Schwefel's problem 1.2	$[-100, 100]^D$	0	Unimodal	Non-separable
f4	Schwefel's problem 2.21	$[-100, 100]^D$	0	Unimodal	Separable
f5	Generalized Rosenbrock's functions	$[-30, 30]^D$	0	Multimodal	Partially Separable
f6	Step function	$[-100, 100]^D$	0	Unimodal	Separable
f7	Quartic function	$[-1.28, 1.28]^D$	0	Unimodal	Separable
f8	Generalized Schwefel's problem 2.26	$[-500, 500]^D$	$-418.9829 * D$	Multimodal	Separable
f9	Generalized Rastrigin's function	$[-5.12, 5.12]^D$	0	Multimodal	Separable
f10	Ackley's function	$[-32, 32]^D$	0	Multimodal	Partially Separable
f11	Generalized Griewank function	$[-600, 600]^D$	0	Multimodal	Partially Separable
f12	Generalized Penalized function 1	$[-50, 50]^D$	0	Multimodal	Non-separable
f13	Generalized Penalized function 2	$[-50, 50]^D$	0	Multimodal	Non-separable
Group 2: CEC2015 functions					
$F1_{CEC2015}$	Rotated High Conditioned Elliptic Function	$[-100, 100]^D$	100	Unimodal	Non-separable
$F2_{CEC2015}$	Rotated Cigar Function	$[-100, 100]^D$	200	Unimodal	Non-separable
$F3_{CEC2015}$	Shifted and Rotated Ackley's Function	$-100, 100]^D$	300	Multimodal	Non-separable
$F4_{CEC2015}$	Shifted and Rotated Rastrigin's Function	$[-100, 100]^D$	400	Multimodal	Non-separable
$F5_{CEC2015}$	Shifted and Rotated Schwefel's Function	$[-100, 100]^D$	500	Multimodal	Non-separable
$F6_{CEC2015}$	Hybrid Function 1 (N=3)	$[-100, 100]^D$	600	Hybrid	Non-separable
$F7_{CEC2015}$	Hybrid Function 2 (N=4)	$[-100, 100]^D$	700	Hybrid	Non-separable
$F8_{CEC2015}$	Hybrid Function 3(N=5)	$[-100, 100]^D$	800	Hybrid	Non-separable
$F9_{CEC2015}$	Composition 1 (N=3)	$[-100, 100]^D$	900	Composition	Non-separable
$F10_{CEC2015}$	Composition 2 (N=3)	$[-100, 100]^D$	1000	Composition	Non-separable
$F11_{CEC2015}$	Composition 3 (N=5)	$[-100, 100]^D$	1100	Composition	Non-separable
$F12_{CEC2015}$	Composition 4 (N=5)	$[-100, 100]^D$	1200	Composition	Non-separable
$F13_{CEC2015}$	Composition 5 (N=5)	$[-100, 100]^D$	1300	Composition	Non-separable
$F14_{CEC2015}$	Composition 6 (N=7)	$[-100, 100]^D$	1400	Composition	Non-separable
$F15_{CEC2015}$	Composition 7 (N=10)	$[-100, 100]^D$	1500	Composition	Non-separable

4.1. Comparison with single-strategy-based ABC algorithms

First, we compare sdABC with six well-established single-strategy-based ABC algorithms, i.e.,

(1) Basic ABC [9]

(2) Gbest-guied ABC (GABC) [26]

(3) Modified ABC with modification rate and scaling factor (MABC) [38]

(4) Gaussian bare-bones ABC (GBABC) [41]

(5) Quick ABC (qABC) [10]

(6) ABC with depth-first search framework and elite-guided search equations (DFSABC_elite) [28].

ABC represents the basic ABC algorithm using the original bee updating equation [9]. GABC incorporates the information of global best solution into the search equation to enhance the exploitation ability [26]. MABC uses modification rate and scaling factor to control the frequency and the magnitude of perturbation respectively [38]. GBABC adoptes Gaussian bare-bones search equation for onlooker bees and generalized opposition-based learning strategy for scout bees[41]. qABC employs a new search equation for the onlooker bees which exploits the best solution among neighbors [10]. DFSABC_elite [2] uses a depth-first search framework and two elite-guided search equations to speed up the convergence rate [28]. The parameter settings of these algorithms and our sdABC[3] are presented in Table 2.

[2]DFSABC_elite uses two search equations, but all food sources use the same search equations in both employed and onlooker bee updating phases, therefore DFSABC_elite is still considered as single-strategy-based ABC in this study.

[3]The influence of the minimum selection probability $pa_{\min}$ on sdABC is discussed in Appendix 2.

Table 3 and Table 4 present the results of our sdABC and the six single-strategy-based ABC algorithms for $D = 30$ and $D = 50$ functions, respectively. In both tables, Wilcoxons rank sum test at a 0.05 significance level is conducted between our sdABC and the six ABC algorithms. The best result in terms of mean error values are highlighted in boldface.

From Table 3, when $D = 30$, our sdABC performs better than the six single-strategy-based ABC algorithms on both the conventional functions and CEC2015 functions. To be specific, according to the Wilcoxon's rank sum test, sdABC is significantly better than ABC, GABC, MABC, GBABC, qABC and DFSABC_elite on 18, 17, 25, 20, 19 and 17 functions, respectively. On the other hand, sdABC is worse than ABC, GABC, MABC, GBABC, qABC and DFSABC_elite on 6, 4, 0, 2, 6 and 2 functions, respectively. On the conventional functions, sdABC achieves the best results on 9 functions (i.e., f01, f03, f04, f06, f07, f08, f09, f10, f12), but sdABC is not good for f11 and f13. For CEC2015 functions, sdABC achieves the best results on 11 functions, and qABC achieves the best results on other functions.

From Table 4, when $D = 50$, our sdABC also performs better than the six single-strategy-based ABC algorithms on majority of the benchmark functions, especially on the CEC2015 functions. More specially, sdABC is significantly better than ABC, GABC, MABC, GBABC, qABC and DFSABC_elite on 17, 17, 24, 17, 18 and 12 functions, respectively. On the other hand, sdABC is worse than ABC, GABC, MABC, GBABC, qABC and DFSABC_elite on 9, 9, 2, 6, 9 and 7 functions, respectively. For the conventional functions, DFSABC_elite shows very competitive performance, as it achieves the best results on 7 functions, and sdABC achieves the best results on 5 functions. For the CEC2015 functions, sdABC performs much better than all the six ABC algorithms, and it achieves the best results on 8 functions.

Table 5 presents the rank values of sdABC and the six single-strategy-based ABC algorithms based on Friedman rank test. For both $D = 30$ and $D = 50$ functions, sdABC attains the best rank, qABC the second, followed by DFSABC_elite. In addition, according to the p-value of the post hoc procedures that are given in Table 6, when $D = 30$, sdABC is significantly better than MABC, ABC, GABC, and GBABC, and when $D = 50$, sdABC is significantly better than MABC and GBABC.

Overall, based on the experiments above, we can draw a conclusion that sdABC can achieve better performance than six single-strategy-based ABC algorithms on both the conventional functions and the CEC2015 functions. The superiority of sdABC to the six ABC algorithms becomes more apparent on the complex non-separable CEC2015 functions. qABC and DFSABC_elite are two ABC algorithms which can obtain very good results in several conventional functions, but their performances on the CEC2015 functions are not satisfactory.

The excellent performance of sdABC on non-separable CEC2015 functions should be attributed to the use of differential search strategies, and these strategies can modify more variables each time in updating the candidate solutions. It is also worth noting that sdABC achieves very good performance on the conventional functions with separable characteristics. This is probably because the differential search strategies can adaptively select small values of crossover probability CR, which is beneficial for the optimization of separable functions.

Table 2: Parameter settings for the six single-strategy-based ABC and our sdABC algorithms

Algorithms	Parameter settings
ABC	source number $SN = 50$, $limit = SN \cdot D$
GABC	source number $SN = 50$, $limit = SN \cdot D$, $C = 1.5$
MABC	source number $SN = 50$, $limit = SN \cdot D$, modification rate $MR = 0.4$, scaling factor $SF = 1$
GBABC	source number $SN = 50$, $limit = SN \cdot D$, crossover rate $CR = 0.3$
qABC	source number $SN = 50$, $limit = SN \cdot D$, neighborhood radius $r = 0.3$
DFSABC_elite	source number $SN = 50$, $limit = SN \cdot D$, elite solution proportion $p = 0.1, r = 1/p$
sdABC	source number $SN = 50$, $limit = SN \cdot D$, $pa_{\min} = 0.2$

Table 3: Results of sdABC vs. six single-strategy-based ABC algorithms for $D = 30$ functions

		ABC		GABC		MABC		GBABC		qABC		DFSABC_elite		sdABC
f1	Mean	4.90E-16		3.97E-16		6.20E-16		2.13E-16		4.30E-28		**0.00E+00**		**0.00E+00**
	SD	(7.83E-17)	+	(8.03E-17)	+	(1.59E-16)	+	(3.50E-17)	+	(8.06E-28)	+	**(0.00E+00)**	=	**(0.00E+00)**
f2	Mean	1.17E-15		1.19E-15		7.62E-16		4.46E-16		3.97E-14		**2.42E-87**		3.24E-45
	SD	(1.27E-16)	+	(1.25E-16)	+	(2.64E-16)	+	(6.42E-17)	+	(1.70E-14)	+	**(3.30E-87)**	-	(2.29E-44)
f3	Mean	3.53E+03		3.98E+03		5.07E-02		6.67E+02		2.42E+02		1.89E+03		**9.53E-25**
	SD	(7.75E+02)	+	(1.29E+03)	+	(4.35E-02)	+	(6.59E+02)	+	(1.16E+02)	+	(6.75E+02)	+	**(6.81E-24)**
f4	Mean	1.04E+00		3.98E-01		5.76E+00		1.60E-03		3.24E-01		6.31E-04		**6.24E-18**
	SD	(3.93E-01)	+	(8.27E-02)	+	(1.79E+00)	+	(7.89E-04)	+	(6.78E-02)	+	(1.28E-04)	+	**(2.43E-17)**
f5	Mean	**2.94E-02**		2.00E-01		3.23E+01		1.60E+00		5.43E-01		1.39E+00		5.47E-01
	SD	**(4.23E-02)**	-	(3.54E-01)	-	(3.19E+01)	+	(4.03E+00)	+	(1.04E+00)	-	(2.81E+00)	+	(1.39E+00)
f6	Mean	**0.00E+00**		**0.00E+00**		1.96E-02		**0.00E+00**		**0.00E+00**		**0.00E+00**		**0.00E+00**
	SD	**(0.00E+00)**	=	**(0.00E+00)**	=	(1.40E-01)	=	**(0.00E+00)**	=	**(0.00E+00)**	=	**(0.00E+00)**	=	**(0.00E+00)**
f7	Mean	3.04E-02		1.80E-02		8.41E-02		8.29E-03		2.44E-02		8.20E-03		**1.84E-03**
	SD	(7.02E-03)	+	(4.22E-03)	+	(2.38E-02)	+	(2.13E-03)	+	(6.73E-03)	+	(2.04E-03)	+	**(7.69E-04)**
f8	Mean	3.82E-04		3.82E-04		1.07E+04		1.07E+04		1.07E+04		3.82E-04		**8.98E-05**
	SD	(7.29E-13)	+	(8.97E-13)	+	(9.19E-12)	+	(9.19E-12)	+	(9.19E-12)	+	(2.55E-13)	+	**(1.64E-04)**
f9	Mean	**0.00E+00**		**0.00E+00**		3.97E+01		**0.00E+00**		1.04E-16		**0.00E+00**		**0.00E+00**
	SD	**(0.00E+00)**	=	**(0.00E+00)**	=	(1.64E+01)	+	**(0.00E+00)**	=	(4.22E-16)	=	**(0.00E+00)**	=	**(0.00E+00)**
f10	Mean	3.25E-14		3.05E-14		1.51E+01		2.09E-14		1.05E-03		2.01E-14		**3.55E-15**
	SD	(3.72E-15)	+	(2.48E-15)	+	(6.75E+00)	+	(4.54E-14)	+	(1.13E-03)	+	(2.44E-15)	+	**(0.00E+00)**
f11	Mean	6.97E-17		4.14E-17		1.84E-03		1.52E-17		1.25E-14		**0.00E+00**		3.38E-04
	SD	(1.17E-16)	-	(5.42E-17)	-	(3.90E-03)	+	(3.86E-17)	=	(6.17E-14)	-	**(0.00E+00)**	=	(1.71E-03)
f12	Mean	4.59E-16		3.71E-16		8.13E-03		2.05E-16		4.62E-27		**1.57E-32**		**1.57E-32**
	SD	(8.43E-17)	+	(7.93E-17)	+	(2.81E-02)	+	(2.99E-17)	+	(1.98E-26)	+	**(5.53E-48)**	=	**(5.53E-48)**
f13	Mean	4.38E-16		3.83E-16		4.31E-04		1.93E-16		1.95E-25		**1.35E-32**		2.15E-04
	SD	(7.66E-17)	-	(8.20E-17)	-	(2.15E-03)	+	(4.19E-17)	-	(4.90E-25)	-	**(1.11E-47)**	=	(1.54E-03)
$F1_{CEC2015}$	Mean	2.49E+06		2.71E+06		1.57E+05		2.71E+06		2.02E+06		2.70E+06		**1.93E+03**
	SD	(1.06E+06)	+	(9.89E+05)	+	(9.09E+04)	+	(1.02E+06)	+	(6.67E+05)	+	(1.18E+06)	+	**(5.09E+03)**
$F2_{CEC2015}$	Mean	4.58E+02		3.73E+03		9.49E+02		2.31E+03		4.79E+02		2.63E+03		**0.00E+00**
	SD	(4.60E+02)	+	(3.26E+03)	+	(1.21E+03)	+	(2.26E+03)	+	(5.96E+02)	+	(2.81E+03)	+	**(0.00E+00)**
$F3_{CEC2015}$	Mean	2.01E+01		2.01E+01		2.08E+01		2.02E+01		**2.00E+01**		2.01E+01		2.03E+01
	SD	(1.81E-02)	-	(3.30E-02)	-	(5.73E-02)	+	(2.51E-02)	-	**(1.24E-02)**	-	(3.38E-02)	-	(8.76E-02)
$F4_{CEC2015}$	Mean	8.22E+01		5.38E+01		8.25E+01		5.59E+01		8.24E+01		4.00E+01		**2.29E+01**
	SD	(1.25E+01)	+	(8.50E+00)	+	(1.09E+01)	+	(9.16E+00)	+	(1.32E+01)	+	(6.43E+00)	+	**(4.33E+00)**
$F5_{CEC2015}$	Mean	1.98E+03		1.68E+03		3.54E+03		2.01E+03		1.94E+03		1.67E+03		**1.60E+03**
	SD	(2.51E+02)	+	(2.32E+02)	=	(3.74E+02)	+	(3.30E+02)	+	(1.94E+02)	+	(2.34E+02)	=	**(2.95E+02)**
$F6_{CEC2015}$	Mean	1.04E+06		1.08E+06		1.76E+05		6.19E+05		7.06E+05		8.74E+05		**1.32E+03**
	SD	(5.43E+05)	+	(6.72E+05)	+	(1.14E+05)	+	(2.91E+05)	+	(4.85E+05)	+	(6.03E+05)	+	**(9.28E+02)**
$F7_{CEC2015}$	Mean	6.33E+00		7.12E+00		8.68E+00		8.44E+00		**6.12E+00**		8.23E+00		7.21E+00
	SD	(9.36E-01)	-	(1.25E+00)	=	(2.80E+00)	+	(1.92E+00)	+	**(9.63E-01)**	-	(1.13E+00)	+	(9.64E-01)
$F8_{CEC2015}$	Mean	1.95E+05		2.29E+05		2.01E+04		1.44E+05		1.73E+05		2.43E+05		**3.16E+02**
	SD	(9.48E+04)	+	(1.33E+05)	+	(6.21E+03)	+	(9.21E+04)	+	(9.54E+04)	+	(1.59E+05)	+	**(1.66E+02)**
$F9_{CEC2015}$	Mean	1.04E+02		1.04E+02		1.04E+02		1.03E+02		1.04E+02		1.04E+02		**1.03E+02**
	SD	(2.98E-01)	+	(2.69E-01)	+	(2.82E-01)	+	(1.85E-01)	+	(3.04E-01)	+	(3.28E-01)	+	**(2.32E-01)**
$F10_{CEC2015}$	Mean	5.70E+05		5.63E+05		6.91E+04		2.85E+05		4.82E+05		4.99E+05		**8.92E+02**
	SD	(3.02E+05)	+	(3.70E+05)	+	(4.22E+04)	+	(1.12E+05)	+	(2.26E+05)	+	(2.58E+05)	+	**(2.90E+02)**
$F11_{CEC2015}$	Mean	3.23E+02		3.27E+02		3.28E+02		4.15E+02		**3.16E+02**		3.34E+02		3.89E+02
	SD	(7.53E+00)	=	(7.65E+01)	=	(2.34E+01)	=	(1.50E+02)	=	**(6.22E+00)**	=	(6.74E+01)	=	(9.56E+01)
$F12_{CEC2015}$	Mean	1.06E+02		1.06E+02		1.06E+02		1.05E+02		1.06E+02		1.05E+02		**1.05E+02**
	SD	(4.18E-01)	+	(4.28E-01)	+	(5.90E-01)	+	(4.43E-01)	+	(4.63E-01)	+	(3.51E-01)	+	**(4.55E-01)**
$F13_{CEC2015}$	Mean	3.00E-02		2.73E-02		2.90E-02		2.65E-02		2.95E-02		2.66E-02		**2.59E-02**
	SD	(1.17E-03)	+	(6.75E-04)	+	(1.95E-03)	+	(8.14E-04)	=	(1.26E-03)	+	(4.83E-04)	+	**(1.58E-03)**
$F14_{CEC2015}$	Mean	3.01E+04		3.08E+04		3.32E+04		3.23E+04		**2.89E+04**		3.19E+04		3.15E+04
	SD	(5.94E+03)	-	(4.32E+03)	=	(8.00E+02)	+	(9.30E+02)	+	**(8.31E+03)**	-	(7.96E+02)	+	(5.70E+02)
$F15_{CEC2015}$	Mean	**1.00E+02**		**1.00E+02**		1.00E+02		**1.00E+02**		1.00E+02		**1.00E+02**		**1.00E+02**
	SD	**(1.85E-13)**	=	**(3.59E-13)**	=	(1.57E-12)	=	**(1.42E-13)**	=	(2.38E-09)	+	**(1.88E-13)**	=	**(1.44E-13)**
G1	+/=/-	8-2-3		8-2-3		12-1-0		9-3-1		8-2-3		6-6-1		
G2	+/=/-	10-2-3		9-5-1		13-2-0		11-3-1		11-1-3		11-3-1		
Total	+/=/-	18-4-6		17-7-4		25-3-0		20-6-2		19-3-6		17-9-2		

"+","-", and "=" symbolize a comparison that sdABC is better than, worse than, or similar to its competitor, respectively, according to the Wilcoxon rank sum test at $\alpha = 0.05$.

Table 4: Results of sdABC vs. six single-strategy-based ABC algorithms for $D = 50$ functions

		ABC		GABC		MABC		GBABC		qABC		DFSABC_elite		sdABC
f1	Mean	9.58E-16		8.03E-16		1.78E-15		2.98E-16		7.22E-28		**0.00E+00**		**0.00E+00**
	SD	(1.11E-16)	+	(1.05E-16)	+	(9.84E-16)	+	(3.24E-17)	+	(7.39E-28)	+	**(0.00E+00)**	=	**(0.00E+00)**
f2	Mean	2.20E-15		2.31E-15		1.33E-07		6.82E-16		6.25E-14		**6.50E-87**		6.10E-61
	SD	(1.41E-16)	+	(1.60E-16)	+	(9.48E-07)	+	(5.90E-17)	+	(1.98E-14)	+	**(8.01E-87)**	-	(3.10E-60)
f3	Mean	1.40E+04		1.62E+04		3.07E+00		1.60E+04		1.71E+03		8.41E+03		**4.40E-17**
	SD	(1.71E+03)	+	(3.41E+03)	+	(2.13E+00)	+	(2.89E+03)	+	(5.27E+02)	+	(1.70E+03)	+	**(3.14E-16)**
f4	Mean	6.73E+00		3.27E+00		1.11E+01		3.40E-01		6.12E-01		**3.71E-02**		2.10E+00
	SD	(1.62E+00)	+	(5.29E-01)	+	(1.43E+00)	+	(1.15E-01)	-	(1.60E-01)	-	**(4.90E-03)**	-	(1.50E+00)
f5	Mean	**8.33E-02**		1.95E-01		8.79E+01		1.90E+00		1.10E+00		4.35E+00		5.47E-01
	SD	**(1.47E-01)**	-	(3.76E-01)	-	(5.01E+01)	+	(3.23E+00)	+	(1.72E+00)	+	(1.39E+01)	+	(1.39E+00)
f6	Mean	**0.00E+00**		**0.00E+00**		1.61E+00		**0.00E+00**		**0.00E+00**		**0.00E+00**		3.92E-02
	SD	**(0.00E+00)**	=	**(0.00E+00)**	=	(2.39E+00)	+	**(0.00E+00)**	=	**(0.00E+00)**	=	**(0.00E+00)**	=	(1.96E-01)
f7	Mean	6.75E-02		4.10E-02		2.93E-01		1.66E-02		3.93E-02		1.64E-02		**6.12E-03**
	SD	(1.16E-02)	+	(6.59E-03)	+	(4.69E-01)	+	(4.09E-03)	+	(7.70E-03)	+	(3.19E-03)	+	**(2.63E-03)**
f8	Mean	6.36E-04		6.36E-04		1.78E+04		1.78E+04		1.78E+04		6.36E-04		**1.62E-04**
	SD	(0.00E+00)	+	(7.13E-13)	+	(1.45E+01)	+	(1.10E-11)	+	(1.10E-11)	+	(0.00E+00)	+	**(2.80E-04)**
f9	Mean	**0.00E+00**		**0.00E+00**		2.38E+02		**0.00E+00**		4.53E-16		**0.00E+00**		**0.00E+00**
	SD	**(0.00E+00)**	=	**(0.00E+00)**	=	(7.11E+01)	+	**(0.00E+00)**	=	(1.81E-15)	+	**(0.00E+00)**	=	**(0.00E+00)**
f10	Mean	6.23E-14		**5.61E-14**		1.91E+01		2.27E-02		1.02E-03		3.85E-14		3.45E-02
	SD	(5.38E-15)	-	**(6.07E-15)**	-	(3.80E+00)	+	(1.62E-01)	-	(1.07E-03)	-	(3.36E-15)	-	(1.72E-01)
f11	Mean	7.62E-17		6.10E-17		3.09E-03		**6.53E-18**		9.58E-17		1.45E-04		1.88E-03
	SD	(7.53E-17)	-	(7.79E-17)	-	(5.50E-03)	+	**(2.64E-17)**	-	(5.93E-16)	-	(1.04E-03)	-	(4.58E-03)
f12	Mean	8.79E-16		7.59E-16		1.49E-01		2.82E-16		3.04E-27		**9.42E-33**		2.44E-03
	SD	(1.24E-16)	-	(1.07E-16)	-	(2.85E-01)	+	(4.75E-17)	-	(1.13E-26)	-	**(1.38E-48)**	=	(1.22E-02)
f13	Mean	8.59E-16		7.54E-16		6.25E-01		2.83E-16		4.51E-25		**1.35E-32**		4.87E-30
	SD	(1.09E-16)	+	(9.89E-17)	+	(2.60E+00)	+	(6.06E-17)	+	(1.78E-24)	+	**(1.11E-47)**	=	(3.47E-29)
$F1_{CEC2015}$	Mean	9.74E+06		1.12E+07		7.11E+05		1.37E+07		6.49E+06		8.74E+06		**2.89E+04**
	SD	(3.09E+06)	+	(3.66E+06)	+	(4.70E+05)	+	(5.19E+06)	+	(2.53E+06)	+	(3.11E+06)	+	**(4.61E+04)**
$F2_{CEC2015}$	Mean	1.49E+03		6.64E+03		4.36E+03		1.01E+04		1.73E+03		6.40E+03		**0.00E+00**
	SD	(1.30E+03)	+	(6.93E+03)	+	(2.89E+03)	+	(1.15E+04)	+	(1.69E+03)	+	(7.46E+03)	+	**(0.00E+00)**
$F3_{CEC2015}$	Mean	2.01E+01		2.02E+01		2.10E+01		2.02E+01		**2.00E+01**		2.00E+01		2.04E+01
	SD	(1.57E-02)	-	(3.42E-02)	-	(3.57E-02)	+	(2.22E-02)	-	**(9.70E-03)**	-	(2.24E-02)	-	(4.99E-02)
$F4_{CEC2015}$	Mean	2.12E+02		1.38E+02		2.37E+02		1.46E+02		2.12E+02		1.00E+02		**5.75E+01**
	SD	(2.50E+01)	+	(1.26E+01)	+	(2.11E+01)	+	(1.23E+01)	+	(2.13E+01)	+	(1.14E+01)	+	**(1.09E+01)**
$F5_{CEC2015}$	Mean	3.77E+03		3.45E+03		7.34E+03		3.92E+03		3.69E+03		**3.15E+03**		3.22E+03
	SD	(3.49E+02)	+	(4.38E+02)	+	(4.35E+02)	+	(3.41E+02)	+	(3.77E+02)	+	**(2.96E+02)**	=	(3.40E+02)
$F6_{CEC2015}$	Mean	2.13E+06		2.47E+06		5.76E+05		2.97E+06		1.50E+06		1.83E+06		**1.59E+04**
	SD	(7.23E+05)	+	(1.00E+06)	+	(2.03E+05)	+	(1.15E+06)	+	(7.77E+05)	+	(7.87E+05)	+	**(3.51E+04)**
$F7_{CEC2015}$	Mean	1.45E+01		1.69E+01		1.63E+01		4.16E+01		**1.45E+01**		2.68E+01		3.98E+01
	SD	(1.53E+00)	-	(5.70E+00)	-	(1.02E+01)	-	(2.14E+01)	=	**(1.54E+00)**	-	(1.56E+01)	=	(1.12E+01)
$F8_{CEC2015}$	Mean	2.17E+06		2.44E+06		**8.25E+05**		2.28E+06		1.62E+06		2.14E+06		2.46E+03
	SD	(7.38E+05)	+	(9.34E+05)	+	**(3.21E+05)**	+	(1.05E+06)	+	(6.73E+05)	+	(8.40E+05)	+	(2.61E+03)
$F9_{CEC2015}$	Mean	1.08E+02		1.07E+02		1.07E+02		1.06E+02		1.07E+02		1.06E+02		**1.05E+02**
	SD	(5.14E-01)	+	(3.34E-01)	+	(6.37E-01)	+	(2.74E-01)	+	(3.95E-01)	+	(3.49E-01)	+	**(3.74E-01)**
$F10_{CEC2015}$	Mean	9.59E+05		1.07E+06		2.24E+04		4.52E+05		7.81E+05		9.26E+05		**1.86E+03**
	SD	(4.10E+05)	+	(4.73E+05)	+	(1.06E+04)	+	(2.64E+05)	+	(4.12E+05)	+	(4.35E+05)	+	**(4.28E+02)**
$F11_{CEC2015}$	Mean	3.44E+02		4.72E+02		6.66E+02		8.45E+02		**3.25E+02**		6.16E+02		5.77E+02
	SD	(1.14E+02)	-	(2.93E+02)	-	(2.15E+02)	=	(3.23E+02)	+	**(1.39E+01)**	-	(3.41E+02)	=	(1.54E+02)
$F12_{CEC2015}$	Mean	**1.09E+02**		1.09E+02		1.15E+02		1.11E+02		1.09E+02		1.25E+02		1.67E+02
	SD	**(7.11E-01)**	-	(8.75E-01)	-	(1.80E+01)	-	(1.28E+01)	-	(8.30E-01)	-	(3.54E+01)	-	(4.50E+01)
$F13_{CEC2015}$	Mean	1.16E-01		9.60E-02		1.05E-01		8.48E-02		1.11E-01		8.89E-02		**8.04E-02**
	SD	(6.20E-03)	+	(3.71E-03)	+	(1.05E-02)	+	(4.49E-03)	+	(6.30E-03)	+	(3.23E-03)	+	**(4.01E-03)**
$F14_{CEC2015}$	Mean	**4.86E+04**		4.96E+04		5.82E+04		6.16E+04		4.95E+04		5.40E+04		6.19E+04
	SD	**(6.84E+03)**	-	(2.59E+02)	-	(1.20E+04)	=	(9.21E+03)	=	(6.99E+00)	-	(7.71E+03)	-	(1.13E+04)
$F15_{CEC2015}$	Mean	1.00E+02		1.00E+02		1.09E+02		1.00E+02		1.00E+02		**1.00E+02**		**1.00E+02**
	SD	(5.87E-01)	+	(1.14E-06)	+	(3.61E+00)	+	(1.99E-13)	=	(2.98E-01)	+	**(2.06E-13)**	=	**(1.24E-13)**
G1	+/=/-	7-2-4		7-2-4		13-0-0		7-2-4		8-1-4		4-5-4		
G2	+/=/-	10-0-5		10-0-5		11-2-2		10-3-2		10-0-5		8-4-3		
Total	+/=/-	17-2-9		17-2-9		24-2-2		17-5-6		18-1-9		12-9-7		

"+","-", and "=" symbolize a comparison that sdABC is better than, worse than, or similar to its competitor, respectively, according to the Wilcoxon rank sum test at $\alpha = 0.05$.

Table 5: Friedman rank of sdABC vs. six single-strategy-based ABC algorithms for all 28 functions

		ABC	GABC	MABC	GBABC	qABC	DFSABC_elite	sdABC
$D = 30$	Friedman rank	4.16	4.16	5.36	4.13	3.73	3.77	2.70
	Final rank	5	6	7	4	2	3	1
$D = 50$	Friedman rank	3.89	4.04	5.46	4.45	3.41	3.52	3.23
	Final rank	4	5	7	6	2	3	1

Table 6: Comparisons of adjusted p-values obtained by Bonferroni-Dunn and Holm procedures for sdABC and six single-strategy-based ABC algorithms for all 28 functions

sdABC	$D = 30$			sdABC	$D = 50$		
vs.	unadjusted p	Bonferroni p	Holm p	vs.	unadjusted p	Bonferroni p	Holm p
MABC	0.0000	0.0000	0.0000	MABC	0.0001	0.0007	0.0007
ABC	0.0112	0.0672	0.0560	GBABC	0.0354	0.2127	0.1772
GABC	0.0112	0.0672	0.0560	GABC	0.1640	0.9839	0.6559
GBABC	0.0133	0.0801	0.0560	ABC	0.2525	1.5148	0.7574
DFSABC_elite	0.0635	0.3809	0.1270	DFSABC_elite	0.6207	3.7241	1.2414
qABC	0.0728	0.4370	0.1270	qABC	0.7571	4.5426	1.2414

4.2. Comparison with multiple-strategy-based ABC and DE algorithms

We compare sdABC with two ABC and four DE algorithms that are built with multiple strategies, i.e.,

(1) Self-adaptive artificial bee colony algorithm based on global best (SABCGB) [35]

(2) Articial bee colony algorithm with variable search strategy (ABCVSS) [33]

(3) Self-adaptive differential evolution (SaDE) [60]

(4) DE with ensemble of mutation strategies and parameters (EPSDE) [61]

(5) Composite differential evolution (CoDE) [56]

(6) Multi-population ensemble differential evolution (MPEDE) [57].

SABCGB and ABCVSS are two multiple-strategy-based ABC algorithms. SABCGB is a multiple-strategy-based ABC using three search strategies based on the global best solution [35]. ABCVSS employs five search strategies to update the solutions, and the success counter values are used to compute the probabilities for selecting different strategies [33]. SaDE, EPSDE, CoDE, and MPEDE are four efficient multiple-strategy-based DE algorithms. SaDE self-adapts both mutation strategies and control parameters through learning from the previous experiences in generating promising solutions [60]. EPSDE employs an ensemble of mutation strategies and control parameters which compete to produce offspring [61]. CoDE combines several effective trial vector generation strategies with some suitable control parameter settings to improve the performance of DE [56]. MPEDE utilizes a multi-population based approach to realize an ensemble of three mutation strategies [57]. The parameter settings of these algorithms are presented in Table 7.

Table 8 and Table 9 presents the results of our sdABC and the multiple-strategy-based ABC and DE algorithms for $D = 30$ and $D = 50$ functions, respectively.

From Table 8, when $D = 30$, sdABC is superior to SABCGB and ABCVSS on majority of the functions according to the Wilcoxons rank sum test. Specifically, sdABC is significantly better than SABCGB and ABCVSS on 19 and 17 functions, while worse on only 1 and 3 functions. Compared with the four multiple-strategy-based DE algorithms, sdABC also achieves very competitive performance. Specifically, sdABC is significantly better than SaDE, EPSDE, CoDE and MPEDE on 20, 13, 10, and 10 functions, while worse on 3, 7, 6 and 7 functions. Among the compared DE algorithms, ESPDE is very efficient for the conventional functions, while MPEDE is very efficient for the CEC2015 functions.

From Table 9, when $D = 50$, sdABC is also superior to SABCGB and ABCVSS. To be specific, sdABC is significantly better than SABCGB and ABCVSS on 16 and 17 functions, while worse only on 5 and 7 functions. Compared with the four multiple-strategy-based DE algorithms, sdABC performs better than SaDE and EPSDE, and similar to CoDE. MPEDE is very efficient for the CEC2015 functions, and sdABC loses the comparison.

Table 10 presents the rank values of sdABC and the six multiple-strategy-based ABC and DE algorithms based on Friedman rank test. When $D = 30$, CoDE attains the best rank, sdABC the second, followed by MPEDE. When $D = 50$, MPEDE attains the best rank, CoDE the second, and sdABC the third. Moreover, according to the p-value of the post hoc procedures that are given in Table 11, for both $D = 30$ and $D = 50$,

there is no significant differences among CoDE, MDEPD and sdABC, but they are significantly better than other multiple-strategy-based ABC and DE algorithms.

In summary, based on the experiments above, we can draw a conclusion that sdABC always performs better than the two multi-strategy-based ABC algorithms on both the conventional functions and the CEC2015 functions. Compared with the multi-strategy-based DE algorithms, sdABC is superior to SaDE and EPSDE, and similar to CoDE and MPEDE.

Table 7: Parameter settings for the multi-strategy-based ABC and DE algorithms

Algorithms	Parameter settings
SABCGB	source number $SN = 50$, $limit = SN \cdot D$, learning period $LP = 50$
ABCVSS	source number $SN = 50$, $limit = SN \cdot D$, number of search equations $NE = 5$
SaDE	population size $NP = 50$, learning period $LP = 50$
EPSDE	population size $NP = 50$
CoDE	population size $NP = 30$
MPEDE	population size $NP = 250$, subpopulation ratio $\lambda = 0.2$, generation gap $ng = 20$

Table 8: Results of sdABC vs. six multi-strategy-based ABC and DE algorithms for $D = 30$ functions

		SABCGB		ABCVSS		SaDE		EPSDE		CoDE		MPEDE		sdABC
f1	Mean	9.93E-86		4.42E-81		**0.00E+00**		**0.00E+00**		3.82E-67		9.93E-45		**0.00E+00**
	SD	(1.43E-85)	+	(1.49E-80)	+	**(0.00E+00)**	=	**(0.00E+00)**	=	(6.88E-67)	+	(5.62E-44)	+	**(0.00E+00)**
f2	Mean	4.42E-43		1.03E-42		3.85E-79		**3.34E-80**		2.55E-35		7.22E-21		3.24E-45
	SD	(2.14E-43)	+	(1.66E-42)	+	(1.10E-78)	-	**(2.10E-79)**	-	(3.51E-35)	+	(1.84E-20)	+	(2.29E-44)
f3	Mean	1.23E+04		4.28E+03		2.31E-07		**1.59E-36**		3.18E-16		3.76E-28		9.53E-25
	SD	(2.37E+03)	+	(1.31E+03)	+	(4.42E-07)	+	**(1.13E-35)**	-	(5.77E-16)	+	(1.65E-27)	-	(6.81E-24)
f4	Mean	2.75E-01		1.79E-01		4.14E-08		2.69E+00		6.57E-16		3.35E-17		**6.24E-18**
	SD	(2.24E-01)	+	(3.45E-02)	+	(2.55E-07)	+	(1.35E+00)	+	(5.76E-16)	+	(3.13E-17)	+	**(2.43E-17)**
f5	Mean	1.03E+00		1.15E-01		2.56E+01		3.13E-01		2.35E-01		**7.82E-02**		5.47E-01
	SD	(2.94E+00)	+	(1.96E-01)	-	(1.90E+01)	+	(1.08E+00)	-	(9.47E-01)	-	**(5.58E-01)**	-	(1.39E+00)
f6	Mean	**0.00E+00**		**0.00E+00**		**0.00E+00**		3.92E-02		**0.00E+00**		**0.00E+00**		**0.00E+00**
	SD	**(0.00E+00)**	=	**(0.00E+00)**	=	**(0.00E+00)**	=	(1.96E-01)	=	**(0.00E+00)**	=	**(0.00E+00)**	=	**(0.00E+00)**
f7	Mean	1.36E-02		1.40E-02		2.52E-03		**9.69E-04**		2.61E-03		9.84E-04		1.84E-03
	SD	(3.86E-03)	+	(3.16E-03)	+	(1.08E-03)	+	**(5.59E-04)**	-	(9.07E-04)	+	(3.19E-04)	-	(7.69E-04)
f8	Mean	3.82E-04		3.82E-04		2.32E+00		**0.00E+00**		3.82E-04		3.82E-04		8.98E-05
	SD	(8.97E-13)	+	(6.32E-13)	+	(1.66E+01)	+	**(0.00E+00)**	-	(0.00E+00)	+	(0.00E+00)	+	(1.64E-04)
f9	Mean	**0.00E+00**		**0.00E+00**		1.95E-02		3.48E-17		**0.00E+00**		**0.00E+00**		**0.00E+00**
	SD	**(0.00E+00)**	=	**(0.00E+00)**	=	(1.39E-01)	=	(2.49E-16)	=	**(0.00E+00)**	=	**(0.00E+00)**	=	**(0.00E+00)**
f10	Mean	2.50E-14		2.35E-14		1.18E-01		4.60E-15		**3.55E-15**		**3.55E-15**		**3.55E-15**
	SD	(4.76E-15)	+	(3.49E-15)	+	(3.29E-01)	+	(1.63E-15)	+	**(0.00E+00)**	=	**(0.00E+00)**	=	**(0.00E+00)**
f11	Mean	**0.00E+00**		**0.00E+00**		2.61E-03		5.31E-04		**0.00E+00**		1.93E-04		3.38E-04
	SD	**(0.00E+00)**	=	**(0.00E+00)**	=	(5.27E-03)	+	(2.76E-03)	=	**(0.00E+00)**	=	(1.38E-03)	=	(1.71E-03)
f12	Mean	**1.57E-32**		**1.57E-32**		1.63E-02		4.07E-03		**1.57E-32**		**1.57E-32**		**1.57E-32**
	SD	**(5.53E-48)**	=	**(5.53E-48)**	=	(6.35E-02)	+	(2.90E-02)	=	**(5.53E-48)**	=	**(5.53E-48)**	=	**(5.53E-48)**
f13	Mean	**1.35E-32**		**1.35E-32**		**1.35E-32**		1.35E-32		**1.35E-32**		**1.35E-32**		2.15E-04
	SD	**(1.11E-47)**	=	**(1.11E-47)**	=	**(1.11E-47)**	=	(1.73E-34)	=	**(1.11E-47)**	=	**(1.11E-47)**	=	(1.54E-03)
$F1_{CEC2015}$	Mean	6.83E+06		2.45E+06		4.14E+05		3.22E+05		1.41E+04		**0.00E+00**		1.93E+03
	SD	(2.29E+06)	+	(9.62E+05)	+	(2.06E+05)	+	(1.43E+06)	+	(9.42E+03)	+	**(0.00E+00)**	-	(5.09E+03)
$F2_{CEC2015}$	Mean	1.05E+04		4.63E+02		1.34E+03		6.96E-10		**0.00E+00**		**0.00E+00**		**0.00E+00**
	SD	(5.58E+03)	+	(5.79E+02)	+	(1.63E+03)	+	(3.64E-09)	=	**(0.00E+00)**	=	**(0.00E+00)**	=	**(0.00E+00)**
$F3_{CEC2015}$	Mean	2.02E+01		2.01E+01		2.05E+01		2.04E+01		**2.01E+01**		2.04E+01		2.03E+01
	SD	(4.12E-02)	-	(2.41E-02)	-	(4.55E-02)	+	(6.33E-02)	+	**(7.28E-02)**	-	(4.98E-02)	+	(8.76E-02)
$F4_{CEC2015}$	Mean	5.32E+01		5.32E+01		3.83E+01		4.58E+01		3.54E+01		2.75E+01		**2.29E+01**
	SD	(7.91E+00)	+	(8.32E+00)	+	(8.38E+00)	+	(8.16E+00)	+	(1.09E+01)	+	(7.39E+00)	+	**(4.33E+00)**
$F5_{CEC2015}$	Mean	1.95E+03		1.84E+03		3.08E+03		3.82E+03		1.83E+03		2.44E+03		**1.60E+03**
	SD	(2.72E+02)	+	(1.94E+02)	+	(7.31E+02)	+	(5.21E+02)	+	(4.97E+02)	+	(3.97E+02)	+	**(2.95E+02)**
$F6_{CEC2015}$	Mean	2.88E+06		7.29E+05		1.23E+04		4.74E+04		2.16E+03		**2.32E+02**		1.32E+03
	SD	(1.63E+06)	+	(4.68E+05)	+	(1.02E+04)	+	(6.90E+04)	+	(2.68E+03)	=	**(1.08E+02)**	-	(9.28E+02)
$F7_{CEC2015}$	Mean	9.42E+00		6.88E+00		6.06E+00		1.25E+01		**2.77E+00**		8.68E+00		7.21E+00
	SD	(1.70E+00)	+	(1.04E+00)	-	(1.18E+00)	-	(1.04E+00)	+	**(5.88E-01)**	-	(1.61E+00)	+	(9.64E-01)
$F8_{CEC2015}$	Mean	5.88E+05		2.16E+05		4.33E+03		7.99E+03		1.53E+02		**4.30E+01**		3.16E+02
	SD	(3.11E+05)	+	(1.12E+05)	+	(4.13E+03)	+	(1.28E+04)	+	(1.09E+02)	-	**(4.04E+01)**	-	(1.66E+02)
$F9_{CEC2015}$	Mean	1.04E+02		1.04E+02		**1.03E+02**		1.05E+02		1.03E+02		1.03E+02		1.03E+02
	SD	(2.99E-01)	+	(3.42E-01)	+	**(1.79E-01)**	-	(1.97E+01)	=	(1.67E-01)	-	(1.57E-01)	+	(2.32E-01)
$F10_{CEC2015}$	Mean	1.28E+06		4.66E+05		9.41E+03		1.15E+04		6.26E+02		**4.23E+02**		8.92E+02
	SD	(7.80E+05)	+	(2.73E+05)	+	(9.17E+03)	+	(1.87E+04)	+	(3.51E+02)	-	**(1.25E+02)**	-	(2.90E+02)
$F11_{CEC2015}$	Mean	3.82E+02		**3.28E+02**		4.37E+02		9.51E+02		4.06E+02		4.11E+02		3.89E+02
	SD	(9.83E+01)	=	**(1.33E+01)**	=	(1.32E+02)	+	(4.52E+01)	+	(7.95E+01)	=	(4.59E+01)	=	(9.56E+01)
$F12_{CEC2015}$	Mean	1.06E+02		1.06E+02		1.05E+02		1.98E+02		1.05E+02		1.05E+02		**1.05E+02**
	SD	(5.01E-01)	+	(4.43E-01)	+	(5.10E-01)	+	(1.06E+01)	+	(5.77E-01)	+	(3.36E-01)	=	**(4.55E-01)**
$F13_{CEC2015}$	Mean	2.62E-02		2.68E-02		2.99E-02		**6.32E-03**		2.61E-02		2.61E-02		2.59E-02
	SD	(2.31E-04)	=	(4.57E-04)	+	(2.56E-03)	+	**(1.23E-04)**	-	(3.91E-04)	=	(4.13E-04)	=	(1.58E-03)
$F14_{CEC2015}$	Mean	3.20E+04		3.14E+04		3.28E+04		**1.01E+04**		3.21E+04		3.24E+04		3.15E+04
	SD	(7.17E+02)	+	(2.12E+02)	=	(8.46E+02)	+	**(1.14E+04)**	-	(1.04E+03)	=	(9.92E+02)	+	(5.70E+02)
$F15_{CEC2015}$	Mean	**1.00E+02**		**1.00E+02**		**1.00E+02**		**1.00E+02**		**1.00E+02**		**1.00E+02**		**1.00E+02**
	SD	**(2.00E-13)**	=	**(2.41E-13)**	=	**(1.43E-13)**	=	**(1.46E-13)**	=	**(1.44E-13)**	=	**(1.44E-13)**	=	**(1.44E-13)**
G1	+/=/-	8-5-0		7-5-1		8-4-1		2-6-5		6-6-1		4-6-3		
G2	+/=/-	11-3-1		10-3-2		12-1-2		10-3-2		4-6-5		6-5-4		
Total	+/=/-	19-8-1		17-8-3		20-5-3		13-8-7		10-12-6		10-11-7		

"+","-", and "=" symbolize a comparison that sdABC is better than, worse than, or similar to its competitor, respectively, according to the Wilcoxon rank sum test at $\alpha = 0.05$.

4.3. Comparison with non-ABC and non-DE algorithms

We compare sdABC with five non-ABC and non-DE algorithms, i.e.,

Table 9: Results of sdABC vs. six multi-strategy-based ABC and DE algorithms for $D = 50$ functions

		SABCGB		ABCVSS		SaDE		EPSDE		CoDE		MPEDE		sdABC
f1	Mean	9.62E-83		6.68E-79		**0.00E+00**		**0.00E+00**		5.02E-80		3.68E-63		**0.00E+00**
	SD	(1.71E-82)	+	(3.03E-78)	+	**(0.00E+00)**	=	**(0.00E+00)**	=	(9.99E-80)	+	(1.50E-62)	+	**(0.00E+00)**
f2	Mean	2.86E-41		5.06E-41		1.68E-80		**0.00E+00**		8.36E-43		2.26E-27		6.10E-61
	SD	(1.01E-41)	+	(1.01E-40)	+	(4.33E-80)	-	**(0.00E+00)**	-	(1.89E-42)	+	(1.14E-26)	+	(3.10E-60)
f3	Mean	3.91E+04		1.74E+04		4.99E-03		**1.59E-18**		1.50E-09		3.36E-14		4.40E-17
	SD	(5.63E+03)	+	(2.27E+03)	+	(6.69E-03)	+	**(1.13E-17)**	-	(3.80E-09)	+	(6.01E-14)	+	(3.14E-16)
f4	Mean	3.07E+00		2.12E+00		3.92E-02		9.73E+00		6.33E-11		**1.98E-13**		2.10E+00
	SD	(1.21E+00)	+	(3.09E-01)	+	(1.84E-01)	-	(3.67E+00)	+	(1.75E-10)	-	**(2.75E-13)**	-	(1.50E+00)
f5	Mean	7.79E-01		5.77E-01		6.57E+01		8.60E-01		**4.16E-01**		7.04E-01		5.47E-01
	SD	(1.21E+00)	+	(2.84E+00)	+	(3.07E+01)	+	(1.66E+00)	+	**(1.19E+00)**	-	(1.53E+00)	+	(1.39E+00)
f6	Mean	**0.00E+00**		**0.00E+00**		**0.00E+00**		1.41E+00		**0.00E+00**		**0.00E+00**		3.92E-02
	SD	**(0.00E+00)**	=	**(0.00E+00)**	=	**(0.00E+00)**	=	(1.65E+00)	+	**(0.00E+00)**	=	**(0.00E+00)**	=	(1.96E-01)
f7	Mean	2.96E-02		2.75E-02		8.75E-03		5.36E-03		4.53E-03		**2.42E-03**		6.12E-03
	SD	(4.17E-03)	+	(4.75E-03)	+	(3.17E-03)	+	(3.89E-03)	-	(1.28E-03)	-	**(7.87E-04)**	-	(2.63E-03)
f8	Mean	6.36E-04		6.36E-04		6.36E-04		**0.00E+00**		4.65E+00		6.36E-04		1.62E-04
	SD	(3.12E-12)	+	(0.00E+00)	+	(0.00E+00)	+	**(0.00E+00)**	-	(2.32E+01)	+	(0.00E+00)	+	(2.80E-04)
f9	Mean	**0.00E+00**		**0.00E+00**		6.63E-01		5.92E-16		8.58E-01		**0.00E+00**		**0.00E+00**
	SD	**(0.00E+00)**	=	**(0.00E+00)**	=	(8.36E-01)	+	(1.05E-15)	+	(8.21E-01)	+	**(0.00E+00)**	=	**(0.00E+00)**
f10	Mean	4.83E-14		4.47E-14		1.14E+00		8.45E-02		**3.55E-15**		6.69E-15		3.45E-02
	SD	(7.25E-15)	-	(3.97E-15)	-	(5.55E-01)	+	(2.97E-01)	+	**(0.00E+00)**	-	(1.16E-15)	=	(1.72E-01)
f11	Mean	**0.00E+00**		**0.00E+00**		1.16E-02		3.19E-03		6.77E-04		8.70E-04		1.88E-03
	SD	**(0.00E+00)**	-	**(0.00E+00)**	-	(3.14E-02)	+	(5.49E-03)	=	(2.37E-03)	=	(2.82E-03)	=	(4.58E-03)
f12	Mean	9.42E-33		9.42E-33		3.42E-02		**1.22E-03**		**1.22E-03**		2.44E-03		2.44E-03
	SD	(1.38E-48)	=	(1.38E-48)	=	(1.12E-01)	+	**(8.71E-03)**	=	**(8.71E-03)**	=	(1.22E-02)	=	(1.22E-02)
f13	Mean	**1.35E-32**		**1.35E-32**		6.28E-04		6.46E-04		2.15E-04		**1.35E-32**		4.87E-30
	SD	**(1.11E-47)**	=	**(1.11E-47)**	=	(3.29E-03)	+	(2.61E-03)	+	(1.54E-03)	=	**(1.11E-47)**	=	(3.47E-29)
$F1_{CEC2015}$	Mean	2.44E+07		9.22E+06		6.81E+05		3.97E+06		2.08E+05		2.90E+04		**2.89E+04**
	SD	(5.21E+06)	+	(3.49E+06)	+	(3.00E+05)	+	(9.91E+06)	+	(1.06E+05)	+	(3.58E+04)	=	**(4.61E+04)**
$F2_{CEC2015}$	Mean	3.21E+04		2.86E+03		4.78E+03		1.46E-08		1.54E-08		**0.00E+00**		**0.00E+00**
	SD	(2.57E+04)	+	(2.23E+03)	+	(4.04E+03)	+	(2.26E-08)	+	(9.39E-08)	+	**(0.00E+00)**	=	**(0.00E+00)**
$F3_{CEC2015}$	Mean	2.02E+01		2.02E+01		2.07E+01		2.06E+01		**2.00E+01**		2.05E+01		2.04E+01
	SD	(3.39E-02)	-	(2.46E-02)	-	(3.57E-02)	+	(8.11E-02)	+	**(5.82E-02)**	-	(4.78E-02)	+	(4.99E-02)
$F4_{CEC2015}$	Mean	1.27E+02		1.31E+02		1.05E+02		1.53E+02		7.65E+01		**5.63E+01**		5.75E+01
	SD	(1.27E+01)	+	(1.49E+01)	+	(1.79E+01)	+	(1.83E+01)	+	(2.01E+01)	+	**(1.34E+01)**	=	(1.09E+01)
$F5_{CEC2015}$	Mean	3.97E+03		3.57E+03		6.02E+03		8.74E+03		3.96E+03		4.71E+03		**3.22E+03**
	SD	(3.26E+02)	+	(3.53E+02)	+	(1.83E+03)	+	(8.24E+02)	+	(7.66E+02)	+	(6.77E+02)	+	**(3.40E+02)**
$F6_{CEC2015}$	Mean	5.99E+06		2.04E+06		8.28E+04		1.74E+05		2.33E+04		**2.02E+03**		1.59E+04
	SD	(1.82E+06)	+	(9.35E+05)	+	(5.15E+04)	+	(1.12E+05)	+	(1.68E+04)	+	**(4.52E+02)**	-	(3.51E+04)
$F7_{CEC2015}$	Mean	3.05E+01		1.51E+01		3.58E+01		2.67E+01		3.85E+01		**1.17E+01**		3.98E+01
	SD	(1.31E+01)	-	(2.23E+00)	-	(2.32E+01)	=	(1.23E+01)	-	(6.62E+00)	-	**(2.46E+00)**	-	(1.12E+01)
$F8_{CEC2015}$	Mean	4.84E+06		2.30E+06		8.36E+04		7.27E+04		7.62E+03		**6.25E+02**		2.46E+03
	SD	(1.78E+06)	+	(8.58E+05)	+	(5.13E+04)	+	(6.14E+04)	+	(5.10E+03)	+	**(3.08E+02)**	-	(2.61E+03)
$F9_{CEC2015}$	Mean	1.07E+02		1.07E+02		1.04E+02		**1.04E+02**		1.04E+02		1.05E+02		1.05E+02
	SD	(4.39E-01)	+	(5.00E-01)	+	(3.13E-01)	-	**(3.15E-01)**	-	(2.88E-01)	-	(3.32E-01)	=	(3.74E-01)
$F10_{CEC2015}$	Mean	2.64E+06		8.99E+05		1.68E+04		3.36E+03		1.27E+03		**1.10E+03**		1.86E+03
	SD	(1.20E+06)	+	(4.58E+05)	+	(2.14E+04)	+	(2.69E+03)	+	(2.72E+02)	-	**(2.21E+02)**	-	(4.28E+02)
$F11_{CEC2015}$	Mean	7.33E+02		**4.25E+02**		8.65E+02		1.63E+03		5.55E+02		5.33E+02		5.77E+02
	SD	(4.09E+02)	=	**(2.47E+02)**	-	(1.16E+02)	+	(6.66E+01)	+	(6.50E+01)	-	(4.52E+01)	-	(1.54E+02)
$F12_{CEC2015}$	Mean	1.20E+02		**1.08E+02**		1.18E+02		2.00E+02		1.29E+02		1.42E+02		1.67E+02
	SD	(2.58E+01)	-	**(1.10E+00)**	-	(2.75E+01)	-	(1.00E-04)	+	(3.98E+01)	-	(4.56E+01)	-	(4.50E+01)
$F13_{CEC2015}$	Mean	8.29E-02		9.26E-02		1.06E-01		**1.12E-02**		8.21E-02		8.12E-02		8.04E-02
	SD	(3.11E-03)	+	(3.41E-03)	+	(9.88E-03)	+	**(1.01E-04)**	-	(4.68E-03)	+	(4.36E-03)	=	(4.01E-03)
$F14_{CEC2015}$	Mean	5.65E+04		4.95E+04		5.61E+04		**1.47E+04**		6.48E+04		6.26E+04		6.19E+04
	SD	(4.44E+03)	=	(1.74E+01)	-	(1.13E+04)	=	**(2.57E+04)**	-	(8.95E+03)	=	(9.74E+03)	=	(1.13E+04)
$F15_{CEC2015}$	Mean	**1.00E+02**		1.00E+02		1.01E+02		1.00E+02		**1.00E+02**		**1.00E+02**		**1.00E+02**
	SD	**(3.48E-13)**	=	(1.03E-03)	+	(5.46E-01)	+	(6.01E-02)	+	**(1.44E-13)**	=	**(1.23E-13)**	=	**(1.24E-13)**
G1	+/=/-	7-4-2		7-4-2		9-2-2		6-3-4		5-4-4		5-6-2		
G2	+/=/-	9-3-3		10-0-5		11-2-2		11-0-4		7-2-6		2-7-6		
Total	+/=/-	16-7-5		17-4-7		20-4-4		17-3-8		12-6-10		7-13-8		

"+","-", and "=" symbolize a comparison that sdABC is better than, worse than, or similar to its competitor, respectively, according to the Wilcoxon rank sum test at $\alpha = 0.05$.

Table 10: Friedman rank of sdABC vs. six multi-strategy-based ABC and DE algorithms for all 28 functions

		SABCGB	ABCVSS	SaDE	EPSDE	CoDE	MPEDE	sdABC
$D = 30$	Friedman rank	4.95	4.20	5.00	4.48	3.05	3.23	3.09
	Final rank	6	4	7	5	1	3	2
$D = 50$	Friedman rank	4.57	3.91	5.09	4.54	3.45	2.96	3.48
	Final rank	6	4	7	5	2	1	3

Table 11: Comparisons of adjusted p-values obtained by Bonferroni-Dunn and Holm procedures for sdABC and six multi-strategy-based ABC and DE algorithms for all 28 functions.

CoDE	$D = 30$			MPEDE	$D = 50$		
vs.	unadjusted p	Bonferroni p	Holm p	vs.	unadjusted p	Bonferroni p	Holm p
SaDE	0.0007	0.0045	0.0045	SaDE	0.0002	0.0014	0.0014
SABCGB	0.0010	0.0063	0.0052	SABCGB	0.0054	0.0323	0.0269
EPSDE	0.0133	0.0801	0.0534	EPSDE	0.0065	0.0390	0.0269
ABCVSS	0.0478	0.2866	0.1433	ABCVSS	0.1012	0.6070	0.3035
MPEDE	0.7571	4.5426	1.5142	sdABC	0.3697	2.2185	0.7395
sdABC	0.9507	5.7041	1.5142	CoDE	0.4037	2.4220	0.7395

(1) Teaching-learning-based optimization (TLBO) [62]

(2) Nonhomogeneous cuckoo search algorithm (NoCuSa) [63]

(3) Loser-out tournament based fireworks algorithm (LoTFWA) [64]

(4) Global-best brain storm optimization (GBSO) [65]

(5) Ensemble particle swarm optimization (EPSO) [66].

TLBO is a new evolutionary algorithm simulating teaching and learning procedure in a class [62]. NoCuSa is an improved cuckoo search algorithm with nonhomogeneous search strategies based on quantum mechanism [63]. LoTFWA is a state-of-the-art fireworks algorithm in which fireworks compete with each other and the losers will be forced to restart from a new location [64]. GBSO is a global-best version of brain storm optimization with a fitness based grouping mechanism [65]. EPSO is multiple-strategy-based PSO algorithm evolved by an ensemble of five PSO strategies [66]. The parameter settings of these algorithms are presented in Table 12.

Table 13 and Table 14 present the results of our sdABC and the non-ABC and non-DE algorithms for $D = 30$ and $D = 50$ functions, respectively. From Table 13, when $D = 30$, sdABC performs better than five non-ABC and non-DE algorithms on majority of the functions. To be specific, according to the Wilcoxons rank sum test, sdABC is significantly better than TLBO, NoCuSa, LoTFWA, GBSO and EPSO on 20, 26, 22, 20 and 18 functions, while worse on 4, 0, 3, 6 and 2 functions, respectively. From Table 14, when $D = 50$, sdABC also performs better than five non-ABC and non-DE algorithms on majority of the functions. Specially, sdABC is significantly better than TLBO, NoCuSa, LoTFWA, GBSO and EPSO on 18, 26, 18, 27 and 15 functions, while worse on 8, 1, 6, 8 and 8 functions, respectively.

Table 15 presents the rank values of sdABC and the non-ABC and non-DE algorithms based on Friedman rank test. When $D = 30$, sdABC attains the best rank, EPSO the second, followed by GBSO. When $D = 50$, EPSO attains the best rank, sdABC the second, followed by GBSO. Moreover, according to the p-value of the post hoc procedures that are given in Table 16, sdABC and EPSO are significantly better than the other algorithms.

Based on the experiments above, we can draw a conclusion that sdABC performs significantly better than TLBO, NoCuSa, LoTFWA and GBSO on both the conventional functions and the CEC2015 functions. Compared with EPSO, sdABC can obtain better results on more functions according to the Wilcoxons rank sum test.

Table 12: Parameter settings for the non-ABC and non-DE algorithms

Algorithms	Parameter settings
TLBO	population size $NP = 40$
NoCuSa	population size $NP = 20$, discover rate $p_a = 0.3$, $\alpha = 1.1$, $\beta = 1.7$, $\delta = 1.6$
LoTFWA	number of fireworks $\mu = 5$, number of explosion sparks $\lambda = 300$, dynamic amplitude coefficients $C_a = 1.2$, $C_r = 0.9$
GBSO	population size $NP = 25$, number of clusters $m = 5$, $p_{one-cluster} = 0.8$, $p_{one-center} = 0.4$, $p_{two-center} = 0.5$
EPSO	population size for the first subgroup $NP_1 = 15$, population size for the second subgroup $NP_2 = 25$

Table 13: Results of sdABC vs. five non-ABC and non-DE algorithms for $D = 30$ functions

		TLBO		NoCuSa		LoTFWA		GBSO		EPSO		sdABC
f1	Mean	**0.00E+00**		9.17E-81		1.18E-22		1.68E-16		2.87E-55		**0.00E+00**
	SD	**(0.00E+00)**	=	(4.73E-80)	+	(2.03E-22)	+	(3.38E-17)	+	(1.40E-54)	+	**(0.00E+00)**
f2	Mean	**0.00E+00**		1.40E-42		3.02E-10		5.12E-09		3.56E-23		3.24E-45
	SD	**(0.00E+00)**	-	(7.18E-42)	+	(9.01E-10)	+	(6.57E-10)	+	(1.43E-22)	+	(2.29E-44)
f3	Mean	**0.00E+00**		2.80E-10		3.07E-04		2.35E-04		2.32E-10		9.53E-25
	SD	**(0.00E+00)**	-	(1.96E-09)	+	(3.13E-04)	+	(1.55E-04)	+	(3.21E-10)	+	(6.81E-24)
f4	Mean	**0.00E+00**		2.91E-03		2.68E-11		1.22E-07		2.17E-05		6.24E-18
	SD	**(0.00E+00)**	-	(3.68E-03)	+	(3.09E-11)	+	(1.80E-07)	+	(2.37E-05)	+	(2.43E-17)
f5	Mean	5.57E-01		1.86E+00		2.51E+01		2.90E+01		5.65E+00		**5.47E-01**
	SD	(2.17E+00)	+	(2.03E+00)	+	(1.19E+01)	+	(2.08E+01)	+	(3.29E+00)	+	**(1.39E+00)**
f6	Mean	**0.00E+00**		7.06E+00		**0.00E+00**		**0.00E+00**		**0.00E+00**		**0.00E+00**
	SD	**(0.00E+00)**	=	(2.48E+01)	+	**(0.00E+00)**	=	**(0.00E+00)**	=	**(0.00E+00)**	=	**(0.00E+00)**
f7	Mean	**1.47E-04**		5.72E-03		2.43E-02		2.23E-03		1.97E-03		1.84E-03
	SD	**(5.45E-05)**	-	(2.96E-03)	+	(7.86E-03)	+	(8.18E-04)	+	(6.97E-04)	=	(7.69E-04)
f8	Mean	3.34E+03		5.77E+03		1.10E+04		3.82E-04		2.55E+01		**8.98E-05**
	SD	(6.41E+02)	+	(1.13E+03)	+	(6.61E+01)	+	(9.10E-13)	+	(5.46E+01)	+	**(1.64E-04)**
f9	Mean	1.13E+01		2.43E+01		3.37E+01		5.99E-02		8.65E-01		**0.00E+00**
	SD	(4.75E+00)	+	(2.18E+01)	+	(8.92E+00)	+	(2.17E-01)	+	(1.18E+00)	+	**(0.00E+00)**
f10	Mean	**3.55E-15**		2.19E+00		2.00E+01		2.83E-09		1.25E-14		**3.55E-15**
	SD	**(0.00E+00)**	=	(9.28E-01)	+	(2.18E-03)	+	(3.83E-10)	+	(3.44E-15)	+	**(0.00E+00)**
f11	Mean	**0.00E+00**		1.85E-02		1.31E-17		8.84E-03		2.90E-04		3.38E-04
	SD	**(0.00E+00)**	=	(2.03E-02)	+	(3.61E-17)	=	(7.57E-03)	+	(1.45E-03)	=	(1.71E-03)
f12	Mean	2.03E-03		2.49E-01		2.65E-15		6.60E-19		**1.57E-32**		**1.57E-32**
	SD	(1.45E-02)	+	(4.93E-01)	+	(3.90E-15)	+	(1.45E-19)	+	**(5.53E-48)**	=	**(5.53E-48)**
f13	Mean	1.65E-02		9.44E-03		4.29E-15		8.85E-18		**1.35E-32**		2.15E-04
	SD	(2.77E-02)	+	(1.05E-02)	+	(6.01E-15)	-	(2.48E-18)	-	**(1.11E-47)**	=	(1.54E-03)
$F1_{CEC2015}$	Mean	1.29E+05		4.29E+04		1.05E+06		8.37E+05		1.12E+05		**1.93E+03**
	SD	(1.15E+05)	+	(2.69E+04)	+	(4.73E+05)	+	(4.15E+05)	+	(6.25E+04)	+	**(5.09E+03)**
$F2_{CEC2015}$	Mean	2.20E+03		3.53E+03		1.71E+02		3.43E+03		1.60E+03		**0.00E+00**
	SD	(2.73E+03)	+	(4.26E+03)	+	(2.94E+02)	+	(3.93E+03)	+	(1.83E+03)	+	**(0.00E+00)**
$F3_{CEC2015}$	Mean	2.09E+01		2.09E+01		**2.00E+01**		2.03E+01		2.05E+01		2.03E+01
	SD	(3.99E-02)	+	(1.53E-01)	+	**(3.16E-05)**	-	(3.14E-01)	=	(5.15E-02)	+	(8.76E-02)
$F4_{CEC2015}$	Mean	8.11E+01		9.58E+01		6.58E+01		**2.13E+01**		4.51E+01		2.29E+01
	SD	(1.53E+01)	+	(3.49E+01)	+	(9.82E+00)	+	**(5.75E+00)**	-	(1.28E+01)	+	(4.33E+00)
$F5_{CEC2015}$	Mean	4.93E+03		5.01E+03		2.25E+03		**5.18E+02**		2.03E+03		1.60E+03
	SD	(1.58E+03)	+	(1.87E+03)	+	(3.40E+02)	+	**(2.65E+02)**	-	(4.56E+02)	+	(2.95E+02)
$F6_{CEC2015}$	Mean	7.48E+04		1.72E+04		3.09E+04		5.30E+04		3.57E+04		**1.32E+03**
	SD	(4.72E+04)	+	(1.48E+04)	+	(2.39E+04)	+	(3.76E+04)	+	(2.78E+04)	+	**(9.28E+02)**
$F7_{CEC2015}$	Mean	9.78E+00		8.62E+00		1.17E+01		6.38E+00		**6.32E+00**		7.21E+00
	SD	(3.28E+00)	+	(4.47E+00)	=	(1.31E+00)	+	(1.07E+00)	-	**(1.95E+00)**	-	(9.64E-01)
$F8_{CEC2015}$	Mean	4.52E+04		1.13E+04		1.87E+04		2.94E+04		2.27E+04		**3.16E+02**
	SD	(3.32E+04)	+	(1.03E+04)	+	(9.36E+03)	+	(1.81E+04)	+	(1.53E+04)	+	**(1.66E+02)**
$F9_{CEC2015}$	Mean	1.03E+02		1.22E+02		1.04E+02		1.14E+02		**1.03E+02**		1.03E+02
	SD	(4.88E-01)	+	(5.67E+01)	+	(1.96E-01)	+	(3.50E+01)	+	**(1.92E-01)**	-	(2.32E-01)
$F10_{CEC2015}$	Mean	4.04E+04		1.01E+04		4.50E+04		3.44E+04		1.73E+04		**8.92E+02**
	SD	(1.86E+04)	+	(7.27E+03)	+	(1.85E+04)	+	(1.91E+04)	+	(7.65E+03)	+	**(2.90E+02)**
$F11_{CEC2015}$	Mean	7.10E+02		1.19E+03		3.05E+02		4.93E+02		**3.03E+02**		3.89E+02
	SD	(2.70E+02)	+	(2.64E+02)	+	(1.42E+00)	=	(4.61E+01)	+	**(1.07E+00)**	=	(9.56E+01)
$F12_{CEC2015}$	Mean	1.17E+02		1.73E+02		1.08E+02		**1.04E+02**		1.05E+02		1.05E+02
	SD	(2.79E+01)	+	(4.48E+01)	+	(9.48E-01)	+	**(5.92E-01)**	-	(5.93E-01)	+	(4.55E-01)
$F13_{CEC2015}$	Mean	3.39E-02		3.71E+00		9.34E-02		2.65E-02		2.71E-02		**2.59E-02**
	SD	(5.39E-03)	+	(8.92E+00)	+	(2.56E-02)	+	(3.20E-04)	+	(9.11E-04)	+	**(1.58E-03)**
$F14_{CEC2015}$	Mean	3.40E+04		3.72E+04		3.15E+04		**3.12E+04**		3.15E+04		3.15E+04
	SD	(1.70E+03)	+	(1.18E+04)	+	(3.29E+02)	-	**(4.23E+02)**	-	(4.48E+02)	=	(5.70E+02)
$F15_{CEC2015}$	Mean	1.01E+02		1.00E+02		1.00E+02		1.00E+02		**1.00E+02**		**1.00E+02**
	SD	(1.63E+00)	+	(5.11E-01)	=	(1.53E-11)	+	(5.45E-10)	+	**(1.79E-13)**	=	**(1.44E-13)**
G1	+/=/-	5-4-4		13-0-0		10-2-1		11-1-1		8-5-0		
G2	+/=/-	15-0-0		13-2-0		12-1-2		9-1-5		10-3-2		
Total	+/=/-	20-4-4		26-2-0		22-3-3		20-2-6		18-8-2		

"+","-", and "=" symbolize a comparison that sdABC is better than, worse than, or similar to its competitor, respectively, according to the Wilcoxon rank sum test at $\alpha = 0.05$.

Table 14: Results of sdABC vs. five non-ABC and non-DE algorithms for $D = 50$ functions

		TLBO		NoCuSa		LoTFWA		GBSO		EPSO		sdABC
f1	Mean	**0.00E+00**		5.41E-58		2.83E-31		1.66E-16		8.33E-45		**0.00E+00**
	SD	**(0.00E+00)**	=	(3.03E-57)	+	(6.02E-31)	+	(3.43E-17)	+	(5.95E-44)	+	**(0.00E+00)**
f2	Mean	**0.00E+00**		2.54E-28		4.31E-02		7.55E-09		3.16E-20		6.10E-61
	SD	**(0.00E+00)**	-	(1.81E-27)	+	(2.41E-01)	+	(8.93E-10)	+	(1.04E-19)	+	(3.10E-60)
f3	Mean	**0.00E+00**		1.02E-04		8.67E-02		1.09E-01		4.02E-05		4.40E-17
	SD	**(0.00E+00)**	-	(2.99E-04)	+	(4.69E-02)	+	(5.13E-02)	+	(3.74E-05)	+	(3.14E-16)
f4	Mean	**0.00E+00**		8.64E-01		3.35E-04		3.69E-05		3.43E-02		2.10E+00
	SD	**(0.00E+00)**	-	(5.02E-01)	-	(1.17E-03)	-	(4.76E-05)	-	(1.33E-02)	-	(1.50E+00)
f5	Mean	1.03E+01		1.28E+01		4.13E+01		6.24E+01		2.23E+01		**5.47E-01**
	SD	(4.45E+00)	+	(2.07E+01)	+	(6.40E-01)	+	(5.17E+01)	+	(1.90E+01)	+	**(1.39E+00)**
f6	Mean	**0.00E+00**		5.97E+01		**0.00E+00**		**0.00E+00**		**0.00E+00**		3.92E-02
	SD	**(0.00E+00)**	=	(9.69E+01)	+	**(0.00E+00)**	=	**(0.00E+00)**	=	**(0.00E+00)**	=	(1.96E-01)
f7	Mean	**1.13E-04**		1.66E-02		5.20E-02		4.08E-03		4.79E-03		6.12E-03
	SD	**(3.90E-05)**	-	(9.00E-03)	+	(1.26E-02)	+	(1.21E-03)	-	(1.28E-03)	-	(2.63E-03)
f8	Mean	6.84E+03		1.09E+04		1.85E+04		1.28E+01		4.88E+01		**1.62E-04**
	SD	(1.04E+03)	+	(2.01E+03)	+	(6.39E+01)	+	(3.43E+01)	+	(8.27E+01)	+	**(2.80E-04)**
f9	Mean	1.24E+01		4.02E+01		6.58E+01		6.14E-01		2.91E+00		**0.00E+00**
	SD	(1.18E+01)	+	(3.05E+01)	+	(1.17E+01)	+	(7.68E-01)	+	(1.53E+00)	+	**(0.00E+00)**
f10	Mean	**3.69E-15**		3.10E+00		2.00E+01		2.25E-09		2.77E-14		3.45E-02
	SD	**(6.96E-16)**	-	(8.45E-01)	+	(9.21E-04)	+	(2.39E-10)	-	(3.98E-15)	-	(1.72E-01)
f11	Mean	**0.00E+00**		9.16E-02		9.36E-17		4.06E-03		3.92E-17		1.88E-03
	SD	**(0.00E+00)**	-	(1.29E-01)	+	(1.07E-16)	-	(6.86E-03)	+	(1.27E-16)	=	(4.58E-03)
f12	Mean	7.89E-31		2.89E-01		2.11E-19		3.81E-19		**9.42E-33**		2.44E-03
	SD	(6.14E-31)	-	(4.96E-01)	+	(2.70E-19)	-	(6.62E-20)	-	**(1.38E-48)**	=	(1.22E-02)
f13	Mean	4.57E-02		1.81E-02		3.54E-19		8.20E-18		**1.35E-32**		4.87E-30
	SD	(7.28E-02)	+	(3.77E-02)	+	(4.00E-19)	+	(1.56E-18)	+	**(1.11E-47)**	=	(3.47E-29)
$F1_{CEC2015}$	Mean	7.07E+05		5.05E+05		5.29E+06		2.86E+06		4.44E+05		**2.89E+04**
	SD	(6.40E+05)	+	(2.35E+05)	+	(1.63E+06)	+	(1.11E+06)	+	(2.04E+05)	+	**(4.61E+04)**
$F2_{CEC2015}$	Mean	5.68E+03		6.41E+03		2.54E+02		7.75E+03		4.12E+03		**0.00E+00**
	SD	(6.12E+03)	+	(8.18E+03)	+	(4.61E+02)	+	(8.88E+03)	+	(4.22E+03)	+	**(0.00E+00)**
$F3_{CEC2015}$	Mean	2.11E+01		2.11E+01		**2.00E+01**		2.09E+01		2.06E+01		2.04E+01
	SD	(4.64E-02)	+	(1.35E-01)	+	**(1.36E-05)**	-	(3.19E-01)	+	(5.35E-02)	+	(4.99E-02)
$F4_{CEC2015}$	Mean	2.05E+02		2.46E+02		1.50E+02		**4.63E+01**		1.03E+02		5.75E+01
	SD	(3.43E+01)	+	(4.87E+01)	+	(2.06E+01)	+	**(1.14E+01)**	-	(2.34E+01)	+	(1.09E+01)
$F5_{CEC2015}$	Mean	8.14E+03		9.90E+03		4.05E+03		**1.53E+03**		4.01E+03		3.22E+03
	SD	(2.88E+03)	+	(3.78E+03)	+	(4.25E+02)	+	**(5.41E+02)**	-	(6.64E+02)	+	(3.40E+02)
$F6_{CEC2015}$	Mean	2.74E+05		5.12E+04		1.50E+05		2.75E+05		1.00E+05		**1.59E+04**
	SD	(1.25E+05)	+	(2.77E+04)	+	(7.25E+04)	+	(1.36E+05)	+	(4.65E+04)	+	**(3.51E+04)**
$F7_{CEC2015}$	Mean	2.96E+01		2.88E+01		2.82E+01		3.65E+01		**2.06E+01**		3.98E+01
	SD	(1.82E+01)	-	(1.64E+01)	=	(1.29E+01)	-	(1.39E+01)	=	**(1.24E+01)**	-	(1.12E+01)
$F8_{CEC2015}$	Mean	2.13E+05		3.31E+04		1.06E+05		1.62E+05		6.36E+04		**2.46E+03**
	SD	(1.45E+05)	+	(2.22E+04)	+	(6.48E+04)	+	(8.14E+04)	+	(3.24E+04)	+	**(2.61E+03)**
$F9_{CEC2015}$	Mean	1.18E+02		1.20E+02		1.07E+02		1.13E+02		**1.04E+02**		1.05E+02
	SD	(5.71E+01)	+	(7.14E+01)	+	(5.08E-01)	+	(3.61E+01)	+	**(2.46E-01)**	-	(3.74E-01)
$F10_{CEC2015}$	Mean	2.76E+03		3.10E+03		8.98E+04		6.66E+03		2.95E+03		**1.86E+03**
	SD	(8.23E+02)	+	(1.67E+03)	+	(2.80E+04)	+	(2.22E+03)	+	(4.78E+02)	+	**(4.28E+02)**
$F11_{CEC2015}$	Mean	1.44E+03		1.85E+03		5.65E+02		4.94E+02		**3.03E+02**		5.77E+02
	SD	(1.12E+02)	+	(3.34E+02)	+	(1.61E+02)	=	(6.93E+01)	-	**(1.02E+00)**	-	(1.54E+02)
$F12_{CEC2015}$	Mean	1.76E+02		2.04E+02		1.12E+02		1.49E+02		**1.08E+02**		1.67E+02
	SD	(4.01E+01)	+	(2.01E+01)	+	(1.41E+00)	-	(4.69E+01)	=	**(7.78E-01)**	-	(4.50E+01)
$F13_{CEC2015}$	Mean	1.47E-01		5.05E+01		6.39E-01		8.78E-02		9.31E-02		**8.04E-02**
	SD	(6.91E-02)	+	(1.13E+02)	+	(1.54E-01)	+	(3.54E-03)	+	(4.19E-03)	+	**(4.01E-03)**
$F14_{CEC2015}$	Mean	6.75E+04		8.76E+04		**4.96E+04**		5.69E+04		5.01E+04		6.19E+04
	SD	(1.08E+04)	+	(3.64E+04)	+	**(4.22E+01)**	=	(8.67E+03)	-	(2.26E+03)	-	(1.13E+04)
$F15_{CEC2015}$	Mean	1.10E+02		1.09E+02		**1.00E+02**		1.00E+02		**1.00E+02**		**1.00E+02**
	SD	(3.55E+00)	+	(4.10E+00)	+	**(1.28E-13)**	=	(2.87E-10)	+	**(2.50E-13)**	=	**(1.24E-13)**
G1	+/=/-	4-2-7		12-0-1		9-1-3		8-1-4		6-4-3		
G2	+/=/-	14-0-1		14-1-0		9-3-3		9-2-4		9-1-5		
Total	+/=/-	18-2-8		26-1-1		18-4-6		27-3-8		15-5-8		

"+","-", and "=" symbolize a comparison that sdABC is better than, worse than, or similar to its competitor, respectively, according to the Wilcoxon rank sum test at $\alpha = 0.05$.

Table 15: Friedman rank of sdABC vs. five non-ABC and non-DE algorithms for all 28 functions

		TLBO	NoCuSa	LoTFWA	GBSO	EPSO	sdABC
$D = 30$	Friedman rank	4.07	4.68	3.86	3.32	2.88	2.20
	Final rank	5	6	4	3	2	1
$D = 50$	Friedman rank	3.77	4.71	3.77	3.48	2.57	2.70
	Final rank	4	6	5	3	1	2

Table 16: Comparisons of adjusted p-values obtained by Bonferroni-Dunn and Holm procedures for sdABC and five non-ABC and non-DE algorithms for all 28 functions.

sdABC	$D = 30$			EPSO	$D = 50$		
vs.	unadjusted p	Bonferroni p	Holm p	vs.	unadjusted p	Bonferroni p	Holm p
NoCuSa	0.0000	0.0000	0.0000	NoCuSa	0.0000	0.0001	0.0001
TLBO	0.0002	0.0009	0.0007	LoTFWA	0.0167	0.0836	0.0669
LoTFWA	0.0009	0.0045	0.0027	TLBO	0.0167	0.0836	0.0669
GBSO	0.0244	0.1222	0.0489	GBSO	0.0685	0.3427	0.1371
EPSO	0.1747	0.8737	0.1747	sdABC	0.8026	4.0129	0.8026

4.4. Comparison with meta-heuristic algorithms in CEC2015

We compared sdABC with some excellent meta-heuristic algorithms which had taken part in the competition on real-parameter single objective optimization at CEC2015, including TEBO [67], dynFWACM [68], SaDPSO [69], ABC-X-LS [70], MVMO [71], and DEsPA [72]. The results of all these algorithms are directly collected from the original papers, and downloaded from the homepage of Prof. P.N. Suganthan (http://www.ntu.edu.sg/home/epnsugan/).

Table 17 and Table 18 presents the results of sdABC and the six meta-heuristic algorithms for $D = 30$ and $D = 50$ functions, respectively. From Table 17, when $D = 30$, sdABC is significantly better than TEBO, dynFWACM SaDPSO, ABC-X-LS, MVMO and DEsPA on 8, 12, 9, 2, 3 and 6 functions, while worse on 6, 1, 3, 11, 10 and 7 functions, respectively. Moreover, sdABC achieves the best results on $F2_{CEC2015}$, $F13_{CEC2015}$ and $F15_{CEC2015}$. From Table 18, when $D = 50$, sdABC is significantly better than TEBO, dynFWACM SaDPSO, ABC-X-LS, MVMO and DEsPA on 10, 12, 5, 2, 1 and 5 functions, while worse on 4, 1, 6, 11, 10 and 9 functions, respectively. Moreover, sdABC still achieves the best results on $F2_{CEC2015}$, $F13_{CEC2015}$ and $F15_{CEC2015}$.

Table 19 presents the rank values of sdABC and five non-ABC and non-DE algorithms based on Friedman rank test. When $D = 30$, MVMO attains the best rank, ABC-X-LS the second, followed by DEsPA and sdABC. When $D = 50$, ABC-X-LS attains the best rank, MVMO the second, followed by DEsPA and sdABC. Moreover, according to the p-value of the post hoc procedures that are given in Table 20, ABC-X-LS, MVMO and DEsPA are significantly better than sdABC, TEBO, dynFWACM and SaDPSO.

Based on the experiments above, it can be concluded that sdABC is inferior to ABC-X-LS, MVMO and DEsPA on CEC2015 functions, while superior to TEBO, dynFWACM and SaDPSO. It is worth noting that sdABC always achieves the best results on $F2_{CEC2015}$, $F13_{CEC2015}$ and $F15_{CEC2015}$ for both $D = 30$ and $D = 50$.

Table 17: Results of sdABC vs. six meta-heuristic algorithms for $D = 30$ functions

		TEBO		dynFWACM		SaDPSO		ABC-X-LS		MVMO		DEsPA		sdABC
$F1_{CEC2015}$	Mean	3.69E+02		6.17E+05		1.93E-02		**0.00E+00**		**0.00E+00**		**0.00E+00**		1.93E+03
	SD	(7.71E+02)	-	(2.49E+05)	+	(8.42E-02)	-	**(0.00E+00)**	-	**(0.00E+00)**	-	**(0.00E+00)**	-	(5.09E+03)
$F2_{CEC2015}$	Mean	4.54E-07		3.31E+03		2.88E+02		**0.00E+00**		**0.00E+00**		**0.00E+00**		**0.00E+00**
	SD	(3.02E-06)	+	(3.59E+03)	+	(1.09E+03)	+	**(0.00E+00)**	=	**(0.00E+00)**	=	**(0.00E+00)**	=	**(0.00E+00)**
$F3_{CEC2015}$	Mean	2.00E+01		2.00E+01		**2.00E+01**		2.00E+01		2.00E+01		2.01E+01		2.03E+01
	SD	(6.32E-02)	-	(5.75E-06)	-	**(4.15E-05)**	-	(1.13E-02)	-	(5.42E-04)	-	(4.36E-02)	-	(8.76E-02)
$F4_{CEC2015}$	Mean	4.41E+01		1.30E+02		4.25E+01		**2.19E+00**		9.54E+00		8.64E+01		2.29E+01
	SD	(1.06E+01)	+	(3.80E+01)	+	(9.31E+00)	+	**(1.35E+00)**	-	(3.54E+00)	-	(2.87E-14)	+	(4.33E+00)
$F5_{CEC2015}$	Mean	1.96E+03		3.38E+03		2.52E+03		**7.78E+01**		1.14E+03		1.85E+03		1.60E+03
	SD	(6.32E+02)	+	(6.98E+02)	+	(3.58E+02)	+	**(1.06E+02)**	-	(2.81E+02)	-	(3.97E+02)	+	(2.95E+02)
$F6_{CEC2015}$	Mean	6.98E+02		2.69E+04		1.38E+03		4.82E+02		3.10E+02		**1.61E+02**		1.32E+03
	SD	(6.52E+02)	-	(1.90E+04)	+	(6.04E+02)	=	(2.20E+02)	-	(1.79E+02)	-	**(8.00E+01)**	-	(9.28E+02)
$F7_{CEC2015}$	Mean	4.42E+00		1.46E+01		9.52E+00		6.48E+00		3.41E+00		**3.09E+00**		7.21E+00
	SD	(1.41E+00)	-	(2.57E+00)	+	(1.93E+00)	+	(2.18E+00)	-	(7.58E-01)	-	**(7.41E-01)**	-	(9.64E-01)
$F8_{CEC2015}$	Mean	1.21E+02		2.40E+04		1.62E+03		1.69E+02		8.14E+01		**2.55E+01**		3.16E+02
	SD	(1.48E+02)	-	(1.32E+04)	+	(1.35E+03)	+	(1.31E+02)	-	(8.39E+01)	-	**(2.29E+01)**	-	(1.66E+02)
$F9_{CEC2015}$	Mean	1.08E+02		1.08E+02		1.03E+02		**1.03E+02**		1.03E+02		1.80E+02		1.03E+02
	SD	(1.22E+00)	+	(9.01E-01)	+	(1.86E-01)	+	**(1.59E-01)**	-	(1.60E-01)	+	(3.60E+01)	+	(2.32E-01)
$F10_{CEC2015}$	Mean	6.21E+02		3.15E+04		6.52E+03		8.86E+02		6.51E+02		**1.71E+02**		8.92E+02
	SD	(9.31E+01)	-	(2.01E+04)	+	(4.66E+03)	+	(1.50E+02)	=	(1.37E+02)	-	**(7.08E+01)**	-	(2.90E+02)
$F11_{CEC2015}$	Mean	4.81E+02		6.72E+02		3.20E+02		3.06E+02		**3.01E+02**		3.11E+02		3.89E+02
	SD	(1.95E+02)	+	(1.54E+02)	+	(8.88E+00)	=	(2.34E+01)	-	**(1.47E-01)**	-	(5.52E+01)	=	(9.56E+01)
$F12_{CEC2015}$	Mean	1.06E+02		1.17E+02		1.05E+02		**1.03E+02**		1.04E+02		1.08E+02		1.05E+02
	SD	(1.04E+00)	+	(1.23E+01)	+	(4.90E-01)	+	**(1.07E+00)**	-	(3.33E-01)	-	(3.19E-01)	+	(4.55E-01)
$F13_{CEC2015}$	Mean	9.87E+01		2.62E-02		1.01E+02		9.43E+01		2.71E-02		8.13E+01		**2.59E-02**
	SD	(5.46E+00)	+	(7.46E-03)	=	(4.06E+00)	+	(5.47E+00)	+	(9.01E-04)	+	(5.60E+00)	+	**(1.58E-03)**
$F14_{CEC2015}$	Mean	3.45E+04		4.49E+04		1.87E+04		**1.40E+04**		3.16E+04		2.81E+04		3.15E+04
	SD	(4.04E+03)	+	(1.02E+03)	+	(5.27E+03)	-	**(1.59E+04)**	-	(3.02E+02)	+	(1.71E+03)	-	(5.70E+02)
$F15_{CEC2015}$	Mean	**1.00E+02**		**1.00E+02**		**1.00E+02**		1.00E+02		1.00E+02		2.73E+02		**1.00E+02**
	SD	**(0.00E+00)**	=	**(0.00E+00)**	=	**(1.19E-13)**	=	(9.66E-09)	+	(0.00E+00)	=	(1.50E-01)	+	**(1.44E-13)**
	+/=/-	8-1-6		12-2-1		9-3-3		2-2-11		3-2-10		6-2-7		

"+","-", and "=" symbolize a comparison that sdABC is better than, worse than, or similar to its competitor, respectively, according to the Wilcoxon rank sum test at $\alpha = 0.05$.

Table 18: Results of sdABC vs. six meta-heuristic algorithms for $D = 50$ functions

		TEBO		dynFWACM		SaDPSO		ABC-X-LS		MVMO		DEsPA		sdABC
$F1_{CEC2015}$	Mean	5.08E+04		1.80E+06		2.78E+02		**0.00E+00**		**0.00E+00**		9.50E+01		2.89E+04
	SD	(3.95E+04)	+	(7.14E+05)	+	(2.88E+02)	-	**(0.00E+00)**	-	**(0.00E+00)**	-	(2.09E+02)	-	(4.61E+04)
$F2_{CEC2015}$	Mean	4.23E+00		5.17E+03		1.96E+02		**0.00E+00**		**0.00E+00**		**0.00E+00**		**0.00E+00**
	SD	(2.94E+01)	+	(8.01E+03)	+	(4.75E+02)	+	**(0.00E+00)**	=	**(0.00E+00)**	=	**(0.00E+00)**	=	**(0.00E+00)**
$F3_{CEC2015}$	Mean	2.00E+01		**2.00E+01**		**2.00E+01**		2.00E+01		**2.00E+01**		2.01E+01		2.04E+01
	SD	(5.35E-02)	-	**(1.40E-06)**	-	**(1.99E-05)**	-	(1.72E-02)	-	**(5.50E-06)**	-	(1.06E-01)	-	(4.99E-02)
$F4_{CEC2015}$	Mean	1.32E+02		2.44E+02		9.12E+01		**6.09E+00**		5.88E+01		1.69E+01		5.75E+01
	SD	(2.53E+01)	+	(5.85E+01)	+	(1.97E+01)	+	**(1.64E+00)**	-	(1.17E+01)	=	(4.31E+00)	-	(1.09E+01)
$F5_{CEC2015}$	Mean	4.51E+03		5.78E+03		4.82E+03		**2.90E+02**		2.83E+03		4.25E+03		3.22E+03
	SD	(8.43E+02)	+	(7.00E+02)	+	(6.17E+02)	+	**(1.54E+02)**	-	(5.34E+02)	-	(8.50E+02)	+	(3.40E+02)
$F6_{CEC2015}$	Mean	1.46E+04		8.81E+04		3.18E+03		1.88E+03		1.41E+03		**9.44E+02**		1.59E+04
	SD	(1.02E+04)	-	(3.82E+04)	+	(8.57E+02)	-	(3.42E+02)	-	(2.97E+02)	-	**(3.25E+02)**	-	(3.51E+04)
$F7_{CEC2015}$	Mean	2.13E+01		5.70E+01		3.48E+01		**1.51E+01**		3.54E+01		4.06E+01		3.98E+01
	SD	(1.60E+01)	-	(2.61E+01)	+	(2.09E+01)	=	**(6.26E+00)**	-	(2.50E+01)	=	(4.94E-01)	+	(1.12E+01)
$F8_{CEC2015}$	Mean	3.14E+03		6.88E+04		2.52E+03		1.27E+03		8.72E+02		**2.93E+02**		2.46E+03
	SD	(1.92E+03)	+	(4.00E+04)	+	(1.50E+03)	=	(4.33E+02)	-	(2.24E+02)	-	**(1.67E+02)**	-	(2.61E+03)
$F9_{CEC2015}$	Mean	1.06E+02		1.11E+02		1.05E+02		**1.04E+02**		1.05E+02		1.90E+02		1.05E+02
	SD	(7.49E-01)	+	(5.26E+01)	+	(2.98E-01)	=	**(3.43E-01)**	-	(1.98E-01)	-	(2.93E+01)	+	(3.74E-01)
$F10_{CEC2015}$	Mean	2.91E+03		4.21E+04		1.79E+04		2.00E+03		**1.43E+03**		5.82E+02		1.86E+03
	SD	(1.09E+03)	+	(1.72E+04)	+	(9.58E+03)	+	(4.13E+02)	=	**(2.75E+02)**	-	(1.09E+02)	-	(4.28E+02)
$F11_{CEC2015}$	Mean	1.15E+03		1.08E+03		3.89E+02		**3.00E+02**		3.26E+02		3.78E+02		5.77E+02
	SD	(9.09E+01)	+	(2.44E+02)	+	(1.01E+02)	-	**(2.15E-04)**	-	(6.04E+01)	-	(4.07E+01)	-	(1.54E+02)
$F12_{CEC2015}$	Mean	1.12E+02		1.75E+02		1.08E+02		**1.04E+02**		1.10E+02		1.08E+02		1.67E+02
	SD	(2.03E+00)	-	(3.80E+01)	+	(4.75E-01)	-	**(9.25E-01)**	-	(1.84E+01)	-	(3.75E-01)	-	(4.50E+01)
$F13_{CEC2015}$	Mean	1.91E+02		1.17E-01		1.93E+02		1.77E+02		1.06E-01		1.48E+02		**8.04E-02**
	SD	(7.16E+00)	+	(4.51E-02)	+	(5.99E+00)	+	(6.88E+00)	+	(1.00E-02)	+	(1.13E+01)	+	**(4.01E-03)**
$F14_{CEC2015}$	Mean	5.39E+04		5.49E+04		**2.78E+04**		4.46E+04		5.08E+04		3.13E+04		6.19E+04
	SD	(9.33E+03)	=	(6.34E+03)	=	**(2.36E+04)**	-	(2.15E+04)	-	(5.16E+03)	-	(5.90E+03)	-	(1.13E+04)
$F15_{CEC2015}$	Mean	1.05E+02		**1.00E+02**		1.00E+02		1.00E+02		**1.00E+02**		2.84E+02		**1.00E+02**
	SD	(2.65E+00)	+	**(0.00E+00)**	=	(1.26E+00)	=	(8.15E-09)	+	**(0.00E+00)**	=	(1.91E-01)	+	**(1.24E-13)**
	+/=/-	10-1-4		12-2-1		5-4-6		2-2-11		1-4-10		5-1-9		

"+","-", and "=" symbolize a comparison that sdABC is better than, worse than, or similar to its competitor, respectively, according to the Wilcoxon rank sum test at $\alpha = 0.05$.

Table 19: Friedman rank of sdABC vs. six meta-heuristic algorithms for 15 CEC2015 functions

		TEBO	dynFWACM	SaDPSO	ABC-X-LS	MVMO	DEsPA	sdABC
$D = 30$	Friedman rank	4.53	6.00	4.53	2.77	2.57	3.63	3.97
	Final rank	5	7	6	2	1	3	4
$D = 50$	Friedman rank	5.27	5.93	4.27	2.40	2.53	3.43	4.17
	Final rank	6	7	5	1	2	3	4

Table 20: omparisons of adjusted p-values obtained by Bonferroni-Dunn and Holm procedures for sdABC and six meta-heuristic algorithms for 15 CEC2015 functions

MVMO	$D = 30$			ABC-X-LS	$D = 50$		
vs.	unadjusted p	Bonferroni p	Holm p	vs.	unadjusted p	Bonferroni p	Holm p
dynFWACM	0.0000	0.0001	0.0001	dynFWACM	0.0000	0.0000	0.0000
SaDPSO	0.0127	0.0760	0.0633	TEBO	0.0003	0.0017	0.0014
TEBO	0.0127	0.0760	0.0633	SaDPSO	0.0180	0.1078	0.0718
sdABC	0.0759	0.4556	0.2278	sdABC	0.0251	0.1507	0.0753
DEsPA	0.1763	1.0578	0.3526	DEsPA	0.1902	1.1412	0.3804
ABC-X-LS	0.7998	4.7991	0.7998	MVMO	0.8658	5.1946	0.8658

4.5. Comparison of time complexity

We evaluate the time complexities of sdABC and the compared ABC algorithms. All the algorithms are implemented using MATLAB and run on an Intel Core i7 CPU (2.5 GHz) and 8GB RAM. The computational complexities of these algorithms for $D = 30$ and $D = 50$ functions are calculated as described in [37], and presented in Tables 21 and 22, respectively. T_0 is a reference as the CPU time cost to run the test program below :

for i = 1 : 1000000

x = 0.55 + (double)i; x = x + x; x = x/2; x = x * x; x = sqrt(x);

x = log(x); x = exp(x); x = x/(x + 2);

end

T_1 is the computation time needed to evaluate $\mathrm{F1}_{\mathrm{CEC2015}}$ with 200,000 function evaluations (FES); $\widehat{T}_2$ is the average CPU time cost for an algorithm to solve $\mathrm{F1}_{\mathrm{CEC2015}}$ with 200,000 FES over 5 runs. $(\widehat{T}_2 - T_1)/T_0$ gives the complexity of an algorithm [37].

From Tables 21 and 22, it can be observed that sdABC has a higher complexity compared with previous ABC algorithms. This is because sdABC employs the multiple differential search strategies and the self-adaptive mechanism to update the selection probabilities, which bring more computational burdens. However, it should be noted that the function evaluation for benchmark function $\mathrm{F1}_{\mathrm{CEC2015}}$ is very cheap. If the function evaluation is costly, the additional overhead caused by multiple differential search strategies and the self-adaptive mechanism will be negligible.

Table 21: Complexity (in seconds) of sdABC and other compared ABC algorithms for $D = 30$ function

Algorithm	T_0	T_1	$\widehat{T}_2$	$(\widehat{T}_2 - T_1)/T_0$
ABC	0.1484	2.3614	12.7220	69.8251
GABC			12.7971	70.3306
MABC			13.6820	76.2949
GBABC			13.2787	73.5767
qABC			17.9466	105.0357
DFSABC_elite			11.1553	59.2660
SABCGB			47.3118	302.9413
ABCVSS			14.4716	81.6163
sdABC			19.9443	118.4990

Table 22: Complexity (in seconds) of sdABC and other compared ABC algorithms for $D = 50$ function

Algorithm	T_0	T_1	$\widehat{T}_2$	$(\widehat{T}_2 - T_1)/T_0$
ABC	0.1484	3.2161	13.5609	69.7186
GABC			13.6390	70.2447
MABC			14.6027	76.7393
GBABC			14.2656	74.4678
qABC			19.2952	108.3648
DFSABC_elite			11.6633	56.9298
SABCGB			48.1733	302.9876
ABCVSS			15.6917	84.0791
sdABC			23.6620	137.7946

4.6. Analysis of the self-adaptive mechanism in sdABC

To gain an in-depth understanding about the search behavior of sdABC, Fig.3 plots the evolution curves of the selection probabilities of different differential strategies and the parameter adaptation on five representative benchmark functions (i.e., functions f1, f9, $\mathrm{F1}_{\mathrm{CEC2015}}$, $\mathrm{F4}_{\mathrm{CEC2015}}$, and $\mathrm{F10}_{\mathrm{CEC2015}}$). f1 is a separable unimodal

function, f9 is a separable multimodal function, $F1_{CEC2015}$ is a non-separable unimodal function, and $F4_{CEC2015}$ and $F10_{CEC2015}$ are two non-separable multimodal functions.

From Fig.3, it can be observed that no single differential strategy can dominate the whole search process on different test functions. Specifically, when solving f1 (Fig.3-a1) and f9 (Fig.3-b1), all three differential strategies have similar selection probabilities. When solving the $F1_{CEC2015}$ (Fig.3-c1) and $F10_{CEC2015}$ (Fig.3-e1), "current-to-rand/1" has the dominating selection probability compared with the other two differential strategies. When solving the $F4_{CEC2015}$ (Fig.3-d1), "current-to-pbest/1 bin" has the largest selection probability. This actually manifests that sdABC can select the suitable differential strategies for different optimization functions.

With regard to the parameter adaptation, we can make the following observations. First, for unimodal functions f1 (Fig.3-a2) and $F1_{CEC2015}$ (Fig.3-c2), μF can reach steady states after the starting stages of optimization. By contrast, for multimodal functions f9 (Fig.3-b2) and $F4_{CEC2015}$ (Fig.3-d2), μF changes over the whole optimization processes. This is probably because the landscape of multimodal functions is very complex compared with unimodal functions, and multimodal functions need to adjust the scale factor F in the whole optimization processes. Second, for two separable functions f1 and f9 (Fig.3-a2 and Fig.3-b2), μCR evolves from 0.5 to a small value. By contrast, for two complex non-separable functions $F4_{CEC2015}$ (Fig.3-d2) and $F10_{CEC2015}$ (Fig.3-e2), μCR evolves from 0.5 to a large value. The reason is that small values of crossover rate CR are useful for the optimization of separable functions, whereas large values of crossover rate CR are useful for the optimization of complex non-separable functions.

Based on the above analysis, we can draw a conclusion that sdABC can select appropriate differential strategies and suitable control parameters by using its self-adaptive mechanism, which is very useful for different kinds of optimization problems.

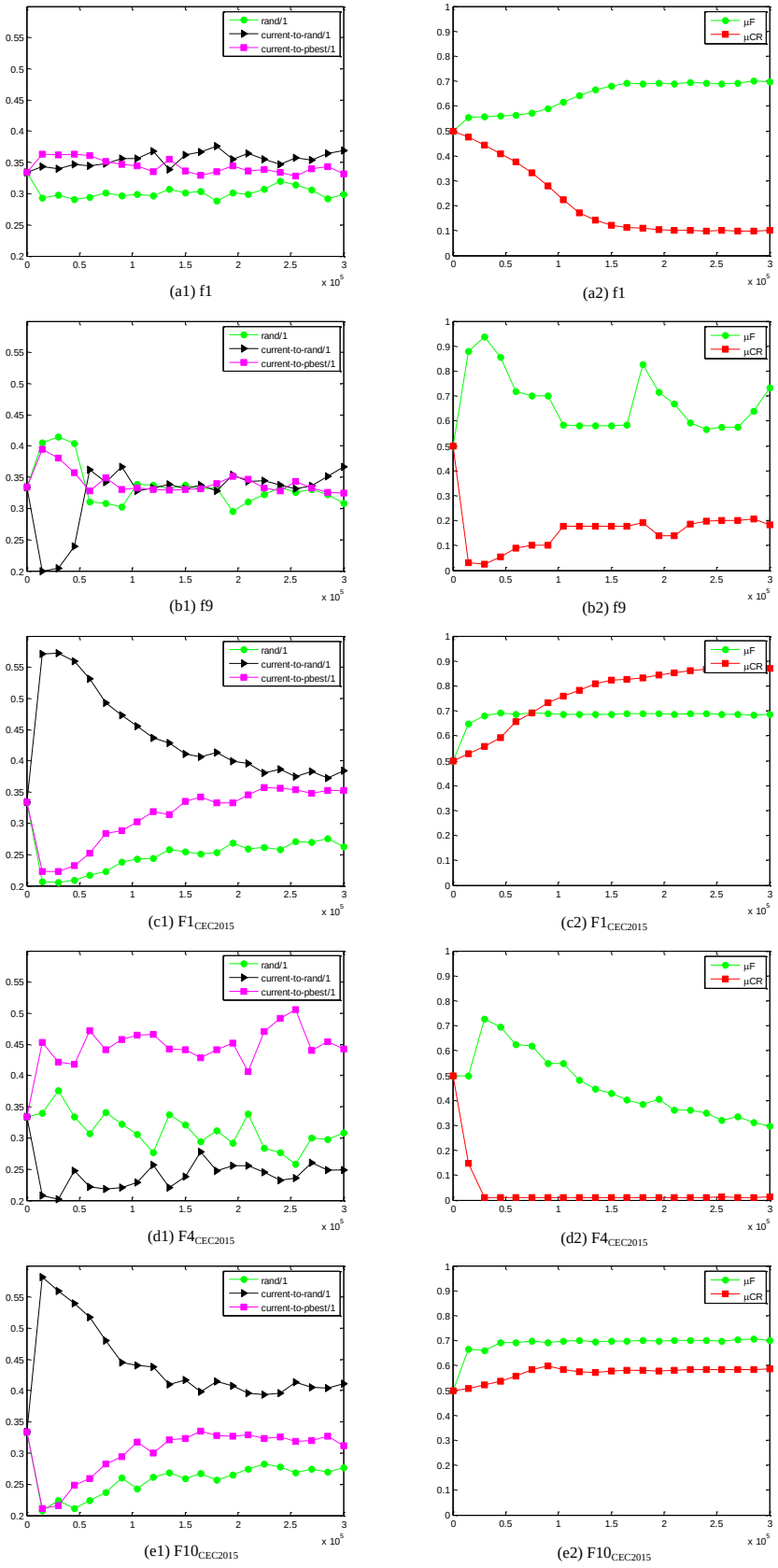


Figure 3: Evolution curves of the selection probabilities of differential search strategies and the parameter adaptation

4.7. Applications to real-world problem

To test the efficiency of our proposed sdABC in dealing with real-world optimization problems, we apply sdABC to the parameter estimation of solar cell models.

4.7.1. Mathematical formulation

Single diode model (SDM) and double diode model (DDM) [73] are two most commonly used models that have been developed to describe the $I-V$ characteristic of solar cells, as illustrated in Fig.4.

The main objective of solar cell parameter estimation problems is to minimize the difference between the experimental data and the simulated ones, so that the optimal values of the unknown model parameters can be extracted. The objective function is often defined as the overall root mean square error (RMSE) [74, 75] as below:

$$\mathrm{RMSE}(\boldsymbol{x}) = \sqrt{\frac{1}{N}\sum_{k=1}^{N} f(V_L, I_L, \boldsymbol{x})^2} \tag{11}$$

where N is the number of experimental data, V_L is the cell output voltage, I_L is the cell output current, and $\boldsymbol{x}$ is solution vector.

In Eq.(11), for the SDM,

$$\begin{cases} f_{SDM}(V_L, I_L, x) = I_{ph} - I_{sd}(exp(\frac{V_L+I_L R_S}{akT/q}) - 1) - \frac{V_L+I_L R_s}{R_{sh}} - I_L \\ x = \{I_{ph}, I_{sd}, R_S, R_{sh}, a\} \end{cases} \tag{12}$$

where I_{ph} is the photo generated current, I_{sd} is reverse saturation current of diode, R_s is the series resistance, R_{sh} denotes the shunt resistance, a is the diode ideality constants, k is the Boltzmann constant ($1.3806503 \times 10^{23} J/K$), T is the temperature of the junction in Kelvin, and q is the electron charge ($1.60217646 \times 10^{-19} C$).

For the DDM,

$$\begin{cases} f_{\mathrm{DDM}}(V_L, I_L, x) = I_{ph} - I_{sd1}(exp(\frac{V_L+I_L R_S}{a_1 kT/q}) - 1) \\ \qquad\qquad - I_{sd2}(exp(\frac{V_L+I_L R_S}{a_2 kT/q}) - 1) - \frac{V_L+I_L R_s}{R_{sh}} - I_L \\ x = \{I_{ph}, I_{sd1}, I_{sd2}, R_S, R_{sh}, a_1, a_2\} \end{cases} \tag{13}$$

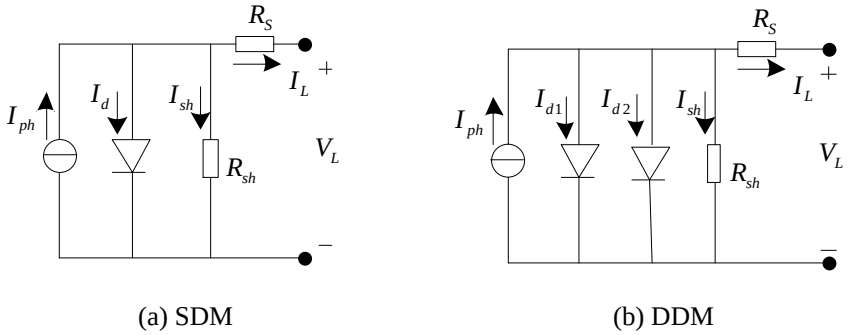
where I_{sd1} and I_{sd2} are the diffusion and saturation currents, a_1 and a_2 denote the diffusion and recombination diode ideality factors, respectively.

4.7.2. Experimental results

We apply sdABC and the other ABC algorithms to solve PVM 752 GaAs thin film cell with single diode and double diode models [76]. The maximum number of function evaluations is set as 20000 for both cases, and each algorithm is run 51 times. Table 23 and 24 present the statistical results of the RMSE values of sdABC and the other ABC algorithms on single diode and double diode models, respectively.

From Table 23, it can be observed that sdABC achieves the best results in terms of minimal, median, mean and maximal RMSE values compared with the other ABC algorithms on single diode model. From Table 24, sdABC also achieves the best results in terms of minimal, median, mean and maximal RMSE values on double

diode model. Therefore, sdABC is a good alterative algorithm for dealing with parameter estimation of solar cell models, and has potential for other real-world optimization problems.



(a) SDM (b) DDM

Figure 4: Solar cell models

Table 23: Statistical results of the RMSE values of sdABC vs. the compared ABC algorithms on single diode model.

Algorithm	RMSE			
	Min	Median	Mean	Max
ABC	2.27770E-03	2.44289E-03	2.41664E-03	2.49186E-03
GABC	2.26856E-03	2.33339E-03	2.33472E-03	2.41138E-03
MABC	2.35817E-03	2.41226E-03	2.40474E-03	2.44027E-03
GBABC	2.26588E-03	2.36781E-03	2.35831E-03	2.44829E-03
qABC	2.29719E-03	2.43336E-03	2.41018E-03	2.47295E-03
DFSABC_elite	2.29208E-03	2.42908E-03	2.40903E-03	2.46746E-03
SABCGB	4.91326E-03	2.53996E-02	2.37862E-02	2.54004E-02
ABCVSS	2.26402E-03	2.39322E-03	2.37470E-03	2.44225E-03
sdABC	**2.26100E-03**	**2.26100E-03**	**2.26100E-03**	**2.26100E-03**

Table 24: Statistical results of the RMSE values of sdABC vs. the compared ABC algorithms on double diode model.

Algorithm	RMSE			
	Min	Median	Mean	Max
ABC	2.41849E-03	2.50078E-03	2.55345E-03	3.05528E-03
GABC	2.29087E-03	2.40934E-03	2.39526E-03	2.45494E-03
MABC	2.37319E-03	2.43849E-03	2.43477E-03	2.45984E-03
GBABC	2.27480E-03	2.37240E-03	2.36549E-03	2.42504E-03
qABC	2.34501E-03	2.46174E-03	2.45875E-03	2.52531E-03
DFSABC_elite	2.29100E-03	2.40377E-03	2.39732E-03	2.46188E-03
SABCGB	8.02563E-03	2.54008E-02	8.43043E-02	9.20475E-01
ABCVSS	2.31041E-03	2.41963E-03	2.41364E-03	2.45572E-03
sdABC	**2.26109E-03**	**2.27262E-03**	**2.28311E-03**	**2.34307E-03**

5. Conclusion

We have developed a new self-adaptive differential ABC (sdABC) algorithm. The proposed algorithm employs multiple differential search strategies in both employed and onlooker bee updating phases. Moreover, a self-adaptive mechanism is used to compute the probabilities for selecting different strategies. The crucial differences between our sdABC and the previous ABC algorithms is that the differential search strategies in the proposed sdABC update more variables based on the combination of mutation and crossover at a time, and this can make sdABC more suitable for the complex non-separable problems.

We have evaluated sdABC on two groups of functions, the conventional functions [59] and the CEC2015 functions [37]. Most of the conventional functions are separable or partially separable, while all of the CEC2015 functions are non-separable. The experimental results show that sdABC can achieve competitive performance

on both the conventional functions and the CEC2015 functions compared with the previous ABC algorithms, and the superiority of sdABC to the previous ABC algorithms becomes more apparent on the non-separable CEC2015 functions. Furthermore, the performance of sdABC is also very competitive compared with DE and some other meta-heuristic algorithms, such as SaDE, EPSDE, CoDE MPEDE, GBSO, and EPSO. The excellent performance of sdABC should be attributed to the use of the self-adaptive differential search strategies, which greatly improves its performance in solving complex non-separable problems.

It is worth noting that although the aim of this research is to enhance the performance of ABC for non-separable problems, the performance of sdABC on separable problems is also very good compared with the previous ABC algorithms. The reason is that the self-adaptive differential search strategies can evolve the crossover rate CR to small values, making the differential search strategies suitable for non-separable problems.

We have also applied sdABC to a real-world problem, i.e., parameter estimation of solar cell models, which is non-separable problem. The experimental results show that sdABC achieves better results than the previous ABC algorithms. Therefore, sdABC has the potential for other real-world optimization problems.

In the future, we think it is interesting to apply the sdABC algorithm for solving complex real-world optimization problems in large-scale power systems and industrial systems.

Acknowledgement

This work was supported in part by the Natural Science Foundation of Jiangsu Province (Grant No. BK 20160540), and National Natural Science Foundation of China (Grant No. 61873114).

Appendix 1. Parameter adaptive technique in sdABC [55, 57]

Let CR_i be the crossover probability of the i-th solution that uses a differential strategy to produce a new solution. At each generation, CR_i is generated according to the following normal distribution:

$$CR_i = randn(\mu CR, 0.1) \tag{14}$$

where μCR is the mean value and 0.1 is the standard deviation value. CR_i will be truncated to [0,1] if necessary. Let S_{CR} be the collection of any CR_i that helps the differential strategy to generate improved solutions. The initial value of μCR is set to 0.5. After each generation, μCR is updated as:

$$\mu CR = (1 - c) \cdot \mu CR + c \cdot mean_A(S_{CR}) \tag{15}$$

where, c is a positive constant between 0 and 1 and $mean_A()$ is a function calculating the arithmetic mean value of elements in S_{CR}.

Similarly, the scaling factor F_i of the i-th solution is updated according to Cauchy distribution as below at each generation:

$$F_i = randc(\mu F, 0.1) \tag{16}$$

where, μF is the location parameter and 0.1 is the scale parameter of the used Cauchy distribution. Also, F_i will be truncated to [0,1] if necessary after the update. Let S_F be the collection of any F_i that helps to generate improved solutions. μF is initialized to 0.5 and updated as below at each generation:

$$\mu F = (1 - c) \cdot \mu F + c \cdot mean_L(S_F) \tag{17}$$

where, $mean_L()$ is the Lehmer mean as below:

$$mean_L(S_F) = \frac{\sum_{F \in S_F} F^2}{\sum_{F \in S_F} F}. \tag{18}$$

Appendix 2. Parameter effect analysis

The proposed sdABC algorithm introduces a new parameter, i.e., minimum selection probability $pa_{\min}$. We analyze the impacts of parameters $pa_{\min}$ on the performance of sdABC. The candidate values for $pa_{\min}$ include 0.05, 0.10, 0.15, 0.2, and 0.25. Fig. 5 plots the sensitivity analysis results of parameters $pa_{\min}$ on some representative functions in terms of mean error values.

It can be observed from Fig. 5 that too small or too large a value for $pa_{\min}$ may deteriorate the performance of sdABC, and $pa_{\min} = 0.2$ is relatively a good choice for the algorithm.

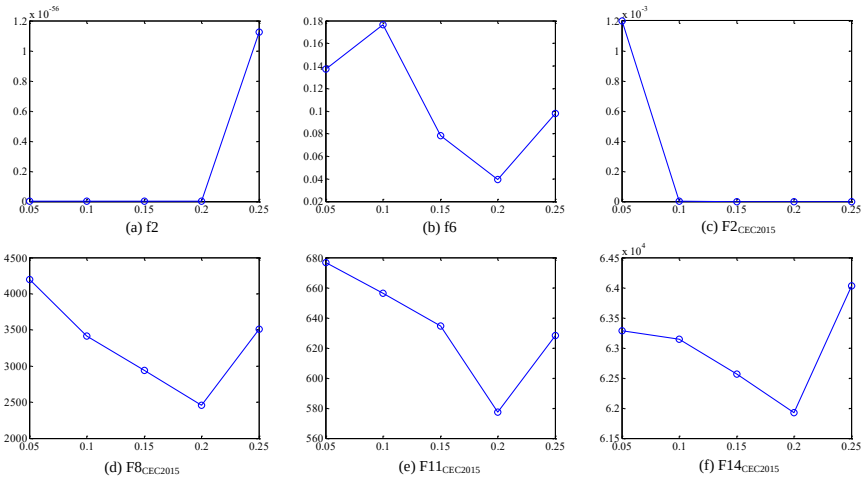


Figure 5: Main effect of the $pa_{\min}$ in sdABC

References

[1] Q. Lin, M. Zhu, G. Li, W. Wang, L. Cui, J. Chen, J. Lu, A novel artificial bee colony algorithm with local and global information interaction, Applied Soft Computing 62 (2018) 702–735.

[2] A. Beck, M. Teboulle, A fast dual proximal gradient algorithm for convex minimization and applications, Operations Research Letters 42 (2014) 1–6.

[3] C. Gu, D. Zhu, Y. Pei, A new inexact sqp algorithm for nonlinear systems of mixed equalities and inequalities, Numerical Algorithms (2017) 1–21.

[4] N. Patel, N. Padhiyar, Modified genetic algorithm using box complex method: Application to optimal control problems, Journal of Process Control 26 (2015) 35–50.

[5] B. Xu, X. Chen, X. Huang, L. Tao, A multistrategy-based multiobjective differential evolution for optimal control in chemical processes, Complexity 2018 (2018).

[6] B. Xu, X. Chen, L. Tao, Differential evolution with adaptive trial vector generation strategy and cluster-replacement-based feasibility rule for constrained optimization, Information Sciences 435 (2018) 240–262.

[7] X. Chen, H. Tianfield, C. Mei, W. Du, G. Liu, Biogeography-based learning particle swarm optimization, Soft Computing 21 (2017) 7519–7541.

[8] X. Chen, B. Xu, W. Du, An improved particle swarm optimization with biogeography-based learning strategy for economic dispatch problems, Complexity 2018 (2018).

[9] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm, Journal of global optimization 39 (2007) 459–471.

[10] D. Karaboga, B. Gorkemli, A quick artificial bee colony (qabc) algorithm and its performance on optimization problems, Applied Soft Computing 23 (2014) 227–238.

[11] X. Chen, H. Tianfield, W. Du, G. Liu, Biogeography-based optimization with covariance matrix based migration, Applied Soft Computing 45 (2016) 71–85.

[12] D. Simon, Biogeography-based optimization, IEEE transactions on evolutionary computation 12 (2008) 702–713.

[13] X. Chen, C. Mei, B. Xu, K. Yu, X. Huang, Quadratic interpolation based teaching-learning-based optimization for chemical dynamic system optimization, Knowledge-Based Systems 145 (2018) 250–263.

[14] K. Yu, X. Chen, X. Wang, Z. Wang, Parameters identification of photovoltaic models using self-adaptive teaching-learning-based optimization, Energy Conversion and Management 145 (2017) 233–246.

[15] Q. Jiang, L. Wang, X. Hei, G. Yu, Y. Lin, The performance comparison of a new version of artificial raindrop algorithm on global numerical optimization, Neurocomputing 179 (2016) 1–25.

[16] A. Rajasekhar, N. Lynn, S. Das, P. N. Suganthan, Computing with the collective intelligence of honey bees–a survey, Swarm and Evolutionary Computation 32 (2017) 25–48.

[17] D. Karaboga, An idea based on honey bee swarm for numerical optimization, Technical Report, Technical report-tr06, Erciyes university, engineering faculty, computer engineering department, 2005.

[18] L. Cui, G. Li, X. Wang, Q. Lin, J. Chen, N. Lu, J. Lu, A ranking-based adaptive artificial bee colony algorithm for global numerical optimization, Information Sciences 417 (2017) 169–185.

[19] J. Luo, Q. Liu, Y. Yang, X. Li, M. Chen, W. Cao, An artificial bee colony algorithm for multi-objective optimisation, Applied Soft Computing 50 (2017) 235–251.

[20] Y. Xiang, Y. Zhou, H. Liu, An elitism based multi-objective artificial bee colony algorithm, European Journal of Operational Research 245 (2015) 168–193.

[21] C. B. Kalayci, A. Hancilar, A. Gungor, S. M. Gupta, Multi-objective fuzzy disassembly line balancing using a hybrid discrete artificial bee colony algorithm, Journal of Manufacturing Systems 37 (2015) 672–682.

[22] C. Ozturk, E. Hancer, D. Karaboga, A novel binary artificial bee colony algorithm based on genetic operators, Information Sciences 297 (2015) 154–170.

[23] X. Yan, Y. Zhu, W. Zou, L. Wang, A new approach for data clustering using hybrid artificial bee colony algorithm, Neurocomputing 97 (2012) 241–250.

[24] X. Chen, B. Xu, C. Mei, Y. Ding, K. Li, Teaching–learning–based artificial bee colony for solar photovoltaic parameter estimation, Applied Energy 212 (2018) 1578–1588.

[25] S. Özyön, D. Aydin, Incremental artificial bee colony with local search to economic dispatch problem with ramp rate limits and prohibited operating zones, Energy Conversion and Management 65 (2013) 397–407.

[26] G. Zhu, S. Kwong, Gbest-guided artificial bee colony algorithm for numerical function optimization, Applied mathematics and computation 217 (2010) 3166–3173.

[27] Z. Liang, K. Hu, Q. Zhu, Z. Zhu, An enhanced artificial bee colony algorithm with adaptive differential operators, Applied Soft Computing 58 (2017) 480–494.

[28] L. Cui, G. Li, Q. Lin, Z. Du, W. Gao, J. Chen, N. Lu, A novel artificial bee colony algorithm with depth-first search framework and elite-guided search equation, Information Sciences 367 (2016) 1012–1044.

[29] W. Gao, S. Liu, L. Huang, A global best artificial bee colony algorithm for global optimization, Journal of Computational and Applied Mathematics 236 (2012) 2741–2753.

[30] W. Gao, S. Liu, L. Huang, A novel artificial bee colony algorithm based on modified search equation and orthogonal learning, IEEE Transactions on Cybernetics 43 (2013) 1011–1024.

[31] S. Cai, W. Long, J. Jiao, Hybridizing artificial bee colony with biogeography-based optimization for constrained mechanical design problems, Journal of Central South University 22 (2015) 2250–2259.

[32] S. M. Chen, A. Sarosh, Y. F. Dong, Simulated annealing based artificial bee colony algorithm for global numerical optimization, Applied mathematics and computation 219 (2012) 3575–3589.

[33] M. S. Kiran, H. Hakli, M. Gunduz, H. Uguz, Artificial bee colony algorithm with variable search strategy for continuous optimization, Information Sciences 300 (2015) 140–157.

[34] H. Wang, Z. Wu, S. Rahnamayan, H. Sun, Y. Liu, J.-s. Pan, Multi-strategy ensemble artificial bee colony algorithm, Information Sciences 279 (2014) 587–603.

[35] Y. Xue, J. Jiang, B. Zhao, T. Ma, A self-adaptive artificial bee colony algorithm based on global best for global optimization, Soft Computing (2017) 1–18.

[36] L. Cui, G. Li, Y. Luo, F. Chen, Z. Ming, N. Lu, J. Lu, An enhanced artificial bee colony algorithm with dual-population framework, Swarm and Evolutionary Computation (2018).

[37] J. Liang, B. Qu, P. Suganthan, Q. Chen, Problem definitions and evaluation criteria for the cec 2015 competition on learning-based real-parameter single objective optimization, Technical Report201411A, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore (2014).

[38] B. Akay, D. Karaboga, A modified artificial bee colony algorithm for real-parameter optimization, Information Sciences 192 (2012) 120–142.

[39] H. Sharma, J. C. Bansal, K. Arya, Opposition based lévy flight artificial bee colony, Memetic Computing 5 (2013) 213–227.

[40] M. S. Kıran, O. Fındık, A directed artificial bee colony algorithm, Applied Soft Computing 26 (2015) 454–462.

[41] X. Zhou, Z. Wu, H. Wang, S. Rahnamayan, Gaussian bare-bones artificial bee colony algorithm, Soft Computing 20 (2016) 907–924.

[42] W. Yu, Z. Zhan, J. Zhang, Artificial bee colony algorithm with an adaptive greedy position update strategy, Soft Computing 22 (2018) 437–451.

[43] X. Song, Q. Yan, M. Zhao, An adaptive artificial bee colony algorithm based on objective function value information, Applied Soft Computing 55 (2017) 384–401.

[44] Y. Xiang, Y. Peng, Y. Zhong, Z. Chen, X. Lu, X. Zhong, A particle swarm inspired multi-elitist artificial bee colony algorithm for real-parameter optimization, Computational Optimization and Applications 57 (2014) 493–516.

[45] S. S. Jadon, R. Tiwari, H. Sharma, J. C. Bansal, Hybrid artificial bee colony algorithm with differential evolution, Applied Soft Computing 58 (2017) 11–24.

[46] B. Wu, C. Qian, W. Ni, S. Fan, Hybrid harmony search and artificial bee colony algorithm for global optimization problems, Computers & Mathematics with Applications 64 (2012) 2621–2634.

[47] C. Zhang, J. Zheng, Y. Zhou, Two modified artificial bee colony algorithms inspired by grenade explosion method, Neurocomputing 151 (2015) 1198–1207.

[48] H. Wang, J. H. Yi, An improved optimization method based on krill herd and artificial bee colony with information exchange, Memetic Computing (2017) 1–22.

[49] W. Gao, L. Huang, S. Liu, F. T. Chan, C. Dai, X. Shan, Artificial bee colony algorithm with multiple search strategies, Applied Mathematics and Computation 271 (2015) 269–287.

[50] F. Harfouchi, H. Habbi, C. Ozturk, D. Karaboga, Modified multiple search cooperative foraging strategy for improved artificial bee colony optimization with robustness analysis, Soft Computing (2017) 1–24.

[51] M. El-Abd, Generalized opposition-based artificial bee colony algorithm, in: Evolutionary Computation (CEC), 2012 IEEE Congress on, IEEE, 2012, pp. 1–4.

[52] S. Das, S. S. Mullick, P. N. Suganthan, Recent advances in differential evolution–an updated survey, Swarm and Evolutionary Computation 27 (2016) 1–30.

[53] G. Wu, X. Shen, H. Li, H. Chen, A. Lin, P. N. Suganthan, Ensemble of differential evolution variants, Information Sciences 423 (2018) 172–186.

[54] S. Das, P. N. Suganthan, Differential evolution: A survey of the state-of-the-art, IEEE transactions on evolutionary computation 15 (2011) 4–31.

[55] J. Zhang, A. C. Sanderson, Jade: adaptive differential evolution with optional external archive, IEEE Transactions on evolutionary computation 13 (2009) 945–958.

[56] Y. Wang, Z. Cai, Q. Zhang, Differential evolution with composite trial vector generation strategies and control parameters, IEEE Transactions on Evolutionary Computation 15 (2011) 55–66.

[57] G. Wu, R. Mallipeddi, P. N. Suganthan, R. Wang, H. Chen, Differential evolution with multi-population based ensemble of mutation strategies, Information Sciences 329 (2016) 329–345.

[58] W.-f. Gao, S.-y. Liu, L.-l. Huang, Enhancing artificial bee colony algorithm using more information-based search equations, Information Sciences 270 (2014) 112–133.

[59] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Transactions on Evolutionary computation 3 (1999) 82–102.

[60] A. K. Qin, V. L. Huang, P. N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE transactions on Evolutionary Computation 13 (2009) 398–417.

[61] R. Mallipeddi, P. N. Suganthan, Q.-K. Pan, M. F. Tasgetiren, Differential evolution algorithm with ensemble of parameters and mutation strategies, Applied soft computing 11 (2011) 1679–1696.

[62] R. V. Rao, V. J. Savsani, D. Vakharia, Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems, Computer-Aided Design 43 (2011) 303–315.

[63] N. J. Cheung, X.-M. Ding, H.-B. Shen, A nonhomogeneous cuckoo search algorithm based on quantum mechanism for real parameter optimization, IEEE transactions on cybernetics 47 (2017) 391–402.

[64] J. Li, Y. Tan, Loser-out tournament based fireworks algorithm for multi-modal function optimization, IEEE Transactions on Evolutionary Computation (2017).

[65] M. El-Abd, Global-best brain storm optimization algorithm, Swarm and Evolutionary Computation 37 (2017) 27–44.

[66] N. Lynn, P. N. Suganthan, Ensemble particle swarm optimizer, Applied Soft Computing 55 (2017) 533–548.

[67] Y.-J. Zheng, X.-B. Wu, Tuning maturity model of ecogeography-based optimization on cec 2015 single-objective optimization test problems, in: Evolutionary Computation (CEC), 2015 IEEE Congress on, IEEE, 2015, pp. 1018–1024.

[68] C. Yu, L. C. Kelley, Y. Tan, Dynamic search fireworks algorithm with covariance mutation for solving the cec 2015 learning based competition problems, in: Evolutionary Computation (CEC), 2015 IEEE Congress on, IEEE, 2015, pp. 1106–1112.

[69] J. Liang, L. Guo, R. Liu, B. Qu, A self-adaptive dynamic particle swarm optimizer, in: Evolutionary Computation (CEC), 2015 IEEE Congress on, IEEE, 2015, pp. 3206–3213.

[70] D. Aydın, T. Sffltzle, A configurable generalized artificial bee colony algorithm with local search strategies, in: Evolutionary Computation (CEC), 2015 IEEE Congress on, IEEE, 2015, pp. 1067–1074.

[71] J. L. Rueda, I. Erlich, Testing mvmo on learning-based real-parameter single objective benchmark optimization problems, in: Evolutionary Computation (CEC), 2015 IEEE Congress on, IEEE, 2015, pp. 1025–1032.

[72] N. Awad, M. Z. Ali, R. G. Reynolds, A differential evolution algorithm with success-based parameter adaptation for cec2015 learning-based optimization, in: Evolutionary Computation (CEC), 2015 IEEE Congress on, IEEE, 2015, pp. 1098–1105.

[73] X. Chen, K. Yu, W. Du, W. Zhao, G. Liu, Parameters identification of solar cell models using generalized oppositional teaching learning based optimization, Energy 99 (2016) 170–180.

[74] K. Yu, J. Liang, B. Qu, Z. Cheng, H. Wang, Multiple learning backtracking search algorithm for estimating parameters of photovoltaic models, Applied Energy 226 (2018) 408–422.

[75] K. Yu, J. Liang, B. Qu, X. Chen, H. Wang, Parameters identification of photovoltaic models using an improved jaya optimization algorithm, Energy Conversion and Management 150 (2017) 742–753.

[76] A. R. Jordehi, Enhanced leader particle swarm optimisation (elpso): An efficient algorithm for parameter estimation of photovoltaic (pv) cells and modules, Solar Energy 159 (2018) 78–87.